Alexandra Alves

Algoritmos para a Determinação da Região Coberta por um k-transmissor

Alexandra Alves

Algoritmos para a Determinação da Região Coberta por um k-transmissor

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Matemática e Aplicações, realizada sob a orientação científica do Doutor Antonio Leslie Bajuelos Domínguez, Professor Auxiliar do Departamento de Matemática da Universidade de Aveiro

o júri / the jury

presidente / president

Doutor António Ferreira Pereira
Professor Auxiliar do Departamento de Matemática da Universidade de Aveiro

vogais / examiners committee

Doutora Ana Mafalda de Oliveira Martins
Investigadora do Centro de Investigação e Desenvolvimento em Matemática e Aplicações do Departamento de Matemática da Universidade de Aveiro

Doutor Antonio Leslie Bajuelos Domínguez
Professor Auxiliar do Departamento de Matemática da Universidade de Aveiro (Orientador)

agradecimentos / acknowledgements

Ao meu orientador, Professor Doutor Antonio Leslie Bajuelos Domínguez, pela disponibilidade, conhecimentos transmitidos, sugestões, excelente orientação e pela confiança demonstradas ao longo do desenvolvimento deste trabalho.

À Doutora Ana Mafalda Martins de Oliveira Martins, pela disponibilidade sempre demonstrada em esclarecer as minha dúvidas.

Ao Professor Gregorio Hernández, pelo importante incentivo e ideias que me deu.

Aos meus pais, pelos valores e ensinamentos que me transmitiram, que me permitiram enfrentar questões para chegar a esta etapa, e pela criação de todas condições para a realização desta dissertação. À minha irmã pelo apoio demonstrado. Obrigada à minha família pela confiança que depositaram em mim.

Ao Rodrigo, por todo o amor e paciência, pelas sugestões que me deu e pela constante motivação, encorajamento e entusiasmo transmitido.

Aos meus colegas e amigos, em especial à Salomé, pelo incentivo e ajuda que me deram.

A todos os meus mais sinceros agradecimentos.

Palavras-chave

Geometria Computacional, Problemas de Visibilidade e Cobertura, Algoritmos de Visibilidade, k-transmissor

Resumo

Os problemas de visibilidade/cobertura têm diversas aplicações a situações reais. Os mais recentes são inspirados nas novas tecnologias, envolvendo dispositivos de rede sem fios. Neste trabalho será abordado o problema de determinar a região coberta pelo sinal emitido por um destes dispositivos, colocado em estruturas geométricas conhecidas como polígonos, uma vez que estes podem representar, de forma apropriada, a geometria de edifícios e são de fácil manipulação computacional, podendo o sinal estudado atravessar um dado número de obstáculos (parades/arestas do polígono), razão pela qual os dispositivos são chamados de k-transmissores, de acordo com o número de obstáculos que o sinal emitido é capaz de atravessar. A resolução deste problema torna-se importante, não só pela sua aplicação directa, mas principalmente porque permitirá prosseguir estudos relacionados, como o problema da determinação do número mínimo de posições para a colocação destes dispositivos num dado polígono, de modo a que os sinais por eles emitidos consigam cobrir a totalidade do polígono.

Nesta dissertação são referidos os algoritmos existentes para a resolução do problema da determinação da região coberta pelo sinal emitido por um dispositivo de redes sem fios, colocado num polígono simples, e são apresentados dois novos algoritmos para a resolução desse mesmo problema.

Com o objectivo de testar e avaliar experimentalmente os algoritmos apresentados e de facilitar o prosseguimento de estudos relacionados, foi desenvolvida uma aplicação que inclui uma interface gráfica para facilitar a introdução de dados de entrada (vértices do polígono e dados relativos aos dispositivos) e a visualização dos resultados (região coberta pelo sinal) por parte do utilizador. Esta aplicação é descrita de forma detalhada nesta dissertação.

Keywords Computational Geometry, Visibility and Coverage Problems, Visibility Algorithms, k-transmitter

Abstract Visibility/coverage problems have several applications to real-life situations. The most recent coverage problems are inspired by new technologies and envolve wireless devices. This work deals with the problem of finding the covered region by the signal of these devices, located on geometrical structures known as polygons, since polygons are appropriate representations of objects geometry and are easily handled by computers. Notice that the signal of those devices can cross a certain number k of walls (edges of the polygon) and that is the reason why the devices are called k-transmitters. The determination of solutions to this problem is important not only because of its direct application but also (and mainly) because it will make possible to proceed related studies, like the problem of find a minimum number of device positions on a given polygon such that these devices collectively cover the whole polygon.

In this dissertation, we refer the existing algorithms to solve the problem of finding the covered region by the signal transmitted by a wireless device, located on a simple polygon, and two new algorithms to also solve this problem are presented.

To experimentally test and evaluate the analyzed algorithms and to become easier to proceed related studies, an application that includes a graphic interface has been developed to make it possible for the user to easily insert data (the vertices of the polygon and the information related to devices) and visualize the final results (the covered region). There is a detailed description of that application in this thesis.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

As novas tecnologias inspiram interessantes problemas geométricos numa área das Ciências da Computação, a Geometria Computacional. Tendo origens na Geometria Euclidiana, esta disciplina estuda algoritmos eficientes para problemas geométricos. É o caso dos problemas de visibilidade[1], como por exemplo, o Problema da Galeria de Arte colocado por Victor Klee, em 1973 [1]: *Quantos guardas são sempre suficientes para cobrir o interior de uma galeria de arte com n paredes?* Em resposta a esta pergunta, Vasek Chvátal [2] provou o chamado Teorema da Galeria de Arte [2]: $\lfloor n/3 \rfloor$ *guardas são ocasionalmente necessários e sempre suficientes para vigiar uma sala com n paredes.*

Os problemas de visibilidade/cobertura têm diversas aplicações a situações reais. Os mais recentes são inspirados nas novas tecnologias, envolvendo dispositivos de rede sem fios (frequentemente referidos como dispositivos wireless). Nesta dissertação, abordar-se-á uma nova variante do Problema da Galeria de Arte, relacionada com estes dispositivos, colocados em estruturas geométricas denominadas de polígonos, uma vez que estes podem representar, de forma apropriada, a geometria de edifícios e são de fácil manipulação computacional. Dado que há interesse em que um computador colocado em qualquer parte de um edifício receba sinal suficientemente forte desses dispositivos, para aceder à internet, colocam-se, então, vários problemas relacionados com estes transmissores de rede sem fios. Uma vez que, em muitos edifícios, a limitação mais significativa para fazer a ligação à rede sem fios prende-se com o número k de paredes (arestas do polígono) que o sinal tem de atravessar (razão pela qual os dispositivos são chamados de k-transmissores) e não tanto com a distância ao transmissor, considera-se, ao longo desta dissertação, que o alcance do sinal é infinito.

Neste trabalho será abordado, especificamente, o problema de determinar a região coberta pelo sinal emitido por um destes dispositivos, colocado num polígono simples. A resolução deste problema torna-se importante, não só pela sua aplicação directa, mas principalmente porque permitirá prosseguir estudos relacionados, como o problema da determinação do número mínimo de posições para a colocação destes dispositivos num dado polígono, de modo a que os sinais por eles emitidos consigam cobrir a totalidade do polí-

[1] A noção de visibilidade é referida frequentemente ao longo deste trabalho também pelo termo *cobertura*, conforme será justificado mais adiante.

gono. Para este último problema, de saber o número mínimo de k-transmissores necessários e sempre suficientes para cobrir um polígono simples P com n vértices, é fácil constatar que, para $k = 0$, a questão é reduzida ao Problema da Galeria de Arte e é $\mathcal{NP}$-difícil [3] e, para $k = n$, apenas um n-transmissor colocado em qualquer ponto de P é suficiente para cobrir P. Para além desta solução trivial, são conhecidos alguns resultados para o caso de polígonos monótonos [4], que serão referidos neste capítulo; contudo, para polígonos arbitrários e polígonos ortogonais, o problema permanece aberto e acredita-se que seja $\mathcal{NP}$-difícil.

Em 2009, durante a abordagem deste problema, A. M. Martins [5] propôs um algoritmo para a resolução do problema de determinar a região coberta pelo sinal emitido por um k-transmissor, que funciona como preparação para a aplicação de meta-heurísticas para a determinação do número mínimo de k-transmissores suficientes para cobrir um polígono simples. Este algoritmo proposto constrói a região de cobertura de um k-transmissor e apresenta complexidade temporal $O(n^2)$. Ao longo desta dissertação, será feita a análise deste algoritmo e a apresentação de novos algoritmos para a construção da região de visibilidade, tendo em conta os seus desempenhos no sentido serem eficientemente utilizados em futuras aplicações.

1.1 Definição do Problema

De seguida, são apresentadas algumas definições essenciais para a formalização do problema que será abordado ao longo deste trabalho.

Definição 1.1. *Uma cadeia poligonal (ou curva poligonal) é definida por um conjunto de n pontos distintos no plano $v_0, v_1, ..., v_{n-1}$, e um conjunto de segmentos de recta $e_0, e_1, ..., e_{n-2}$, onde cada segmento $e_i, i \in 0, 1, ..., n-2$, tem como extremos os pontos v_i e v_{i+1}. Uma cadeia poligonal denota-se por $C = v_0v_1...v_{n-1}$. (ver exemplos da figura 1.1)*

Os pontos $v_0, v_1, ..., v_{n-1}$ são designados vértices e os segmentos $e_0, e_1, ..., e_{n-1}$ de arestas (ver figura 1.1a).

Considera-se uma situação degenerada quando existem 3 vértices consecutivos colineares.

Uma cadeia poligonal diz-se fechada se o último ponto da cadeia é igual ao primeiro, isto é, $v_0 = v_{n-1}$. Considera-se simples se, dadas duas arestas não adjacentes, estas não se intersectarem.

De acordo com o Teorema de Jordan, toda a cadeia plana fechada simples divide o plano em duas regiões: o interior e o exterior da cadeia.

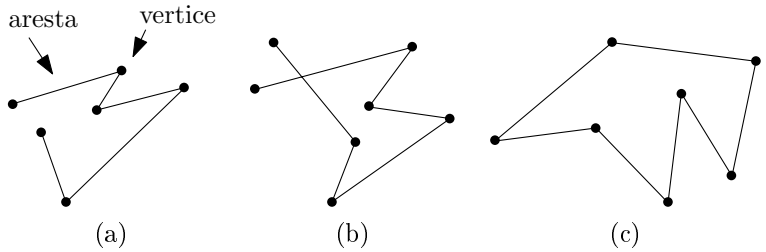


Figura 1.1: Exemplos de cadeias poligonais: (a) Cadeia poligonal simples aberta; (b) Cadeia poligonal não simples aberta; (c) Cadeia poligonal simples fechada.

Definição 1.2. *Designa-se polígono simples o conjunto dos pontos da região interior reunido com os pontos da cadeia poligonal simples fechada. (ver exemplos da figura 1.2)*

Neste trabalho, trabalhar-se-á apenas com polígonos simples, portanto estes serão frequentemente referidos apenas como polígonos ao longo do texto.

Um polígono com n vértices é, por vezes, designado por n-gon.

Denota-se o interior de um polígono P por $int(P)$, o exterior por $ext(P)$ a fronteira por $fr(P)$.

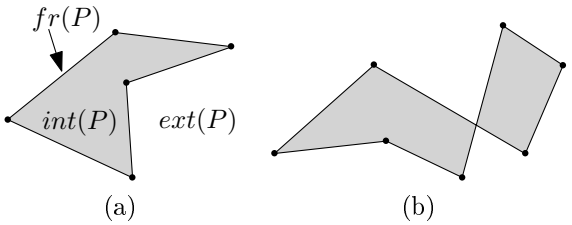


Figura 1.2: Exemplos de polígonos: (a) Polígono simples P (onde se mostra o interior, o exterior e a fronteira de P); (b) Polígono não simples (pois a cadeia poligonal que o define não é simples).

Definição 1.3. *Um polígono P diz-se orientado positivamente (ou que tem orientação horária) se, ao percorrer a fronteira, o interior de P se encontra à esquerda de qualquer aresta orientada v_iv_{i+1}. Analogamente, P tem orientação negativa se o interior de P se encontra à direita de qualquer aresta orientada v_iv_{i+1}. (ver figura 1.3)*

Deste modo, um polígono simples P fica perfeitamente definido por uma conjunto de pontos formados pelos vértices ordenados, quando se percorre a fronteira de P, e uma

orientação, que nos indica onde se situa o interior de P.

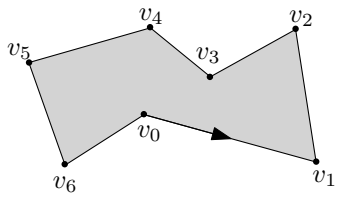


Figura 1.3: Polígono orientado positivamente.

No decorrer deste trabalho, assume-se que todos os polígonos estão positivamente orientados.

Definição 1.4. *Dois pontos p e q, num polígono simples P, dizem-se visíveis (ou p vê q, ou q é visível de p) se o segmento de recta que une p e q não contém pontos do exterior de P. (ver figura 1.4)*

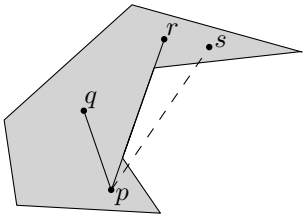


Figura 1.4: Visibilidade: os pontos q e r são visíveis de p, mas s não o é.

Diz-se que uma colecção F de pontos de P cobre (ou ilumina, ou guarda) P se todo o ponto $u \in P$ é visível de um ponto p de F (ver exemplo da figura 1.5).

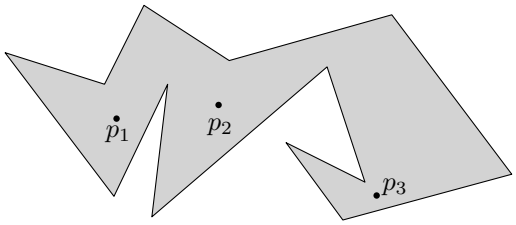


Figura 1.5: $\{p_1, p_2, p_3\}$ cobre o polígono.

É fácil observar que, num polígono convexo P, qualquer ponto de P cobre P. No entanto, na aplicação destas noções à resolução de problemas práticos, verifica-se que os edifícios não apresentam, em geral, formas regulares, pelo que se colocam diversos problemas de iluminação e vigilância.

Além disso, o desenvolvimento da internet e das redes wireless abriu caminho a novas investigações relacionadas com as noções anteriores de visibilidade.

Definição 1.5. *Seja P um polígono com n vértices. Um dispositivo wireless, situado num ponto $x \in P$, que transmite um sinal estável através de, no máximo, k arestas (paredes) de P, em linha recta, é designado por k-transmissor*[2]*.*

No decorrer deste trabalho, por vezes, para simplificar, em vez da expressão "um k-transmissor situado num ponto x", usar-se-á a expressão "x é um k-transmissor".

Nota 1.1. O alcance do sinal do transmissor é considerado infinito, ao longo desta dissertação, pois verifica-se, em muitos casos, que a limitação mais significativa para fazer a ligação à rede sem fios se prende com o número k de paredes (arestas do polígono) que o sinal tem de atravessar e não tanto com a distância ao transmissor. O sinal vai enfraquecendo à medida que atravessa os obstáculos, até que deixa de ser suficientemente forte para se proceder a uma conexão. O valor k de um determinado transmissor representa, portanto, a potência do transmissor, isto é, a capacidade que o sinal, que o transmissor emite, tem de atravessar obstáculos (paredes) até perder força para que seja captado em boas condições por outros dispositivos (como um computador que se pretenda ligar à rede wireless). Deste modo, a noção anterior de visibilidade pode ser estendida agora, tendo em conta a noção de k-transmissor.

Definição 1.6. *Seja P um polígono com n vértices e $k \in \mathbb{N}_0$. Um ponto $y \in P$ é coberto por um k-transmissor situado num ponto $x \in P$ se o segmento de recta xy atravessa, no máximo, k arestas de P (ver figura 1.6).*

Note-se que, pelo facto de a recta xy poder intersectar arestas, o termo k-visibilidade será frequentemente substituído, ao longo deste texto, pelo termo *k-cobertura* (ou simplesmente *cobertura*).

Coloca-se então a questão de saber, dado um polígono P de n vértices, qual o número mínimo de k-transmissores (colocados em pontos de P) necessários para cobrir P. Para abordar este problema, que se acredita ser $\mathcal{NP}$-difícil, faz então sentido desenvolver um algoritmo que determine a região de visibilidade de um k-transmissor colocado num ponto $x \in P$.

Definição 1.7. *Seja P um polígono com n vértices. A região de cobertura de um k-transmissor colocado num ponto $x \in P$ é o conjunto de todos os pontos $y \in \mathbb{R}^2$ que são*

[2]Em notações de trabalhos anteriores, os k-transmissores são também designados por k-modems.

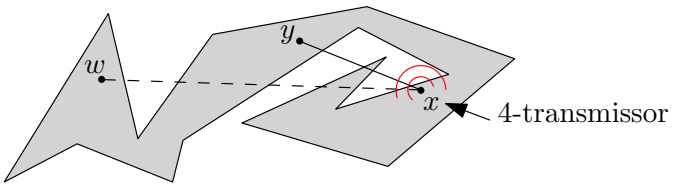


Figura 1.6: Pontos cobertos por transmissor: sendo x um 4-transmissor, x cobre o ponto y, mas não cobre w.

cobertos por x. Este conjunto é denotado por $Vis_k(x, P)$ [3], onde x é o ponto de P onde se situa o k-transmissor (ver figura 1.7).

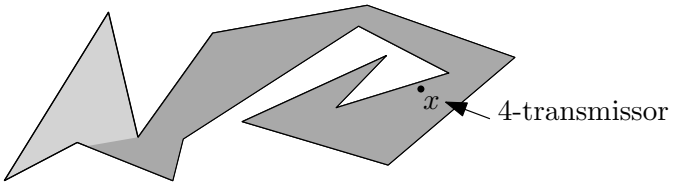


Figura 1.7: Região de visibilidade de um 4-transmissor situado no ponto x (região mais escura).

Importa, então, conhecer algoritmos que determinem a região coberta por um k-transmissor colocado num ponto $x \in P$. Assim, o problema em foco neste trabalho pode ser formalmente definido da seguinte forma:

Dado um polígono simples P e um k-transmissor situado num ponto $x \in P$, pretende-se determinar a região de k-cobertura desse k-transmissor, $Vis_k(x, P)$, de forma eficiente.

É fácil observar que, para $k = 0$, ou seja, para um transmissor cujo sinal não é capaz de atravessa obstáculos, este problema se reduz ao problema clássico de encontrar a região de visibilidade de um ponto, para o qual existe um algoritmo de complexidade temporal linear (Lee, 1983). No entanto, estamos interessados num algoritmo capaz de resolver o problema colocado para qualquer valor de k, com complexidade temporal reduzida.

[3] No caso da visibilidade clássica, este conjunto é frequentemente designado por polígono de visibilidade e é usual denotá-lo simplesmente por $V(x)$.

1.2 Motivações e objectivos gerais

Os problemas de visibilidade são alvo de intensas pesquisas. Mais, concretamente, o problema da determinação da região de cobertura de um k-transmissor apresenta sobretudo interesse prático.

Como já foi referido, a evolução da internet e o seu crescente uso, leva a que surjam cada vez mais dispositivos de rede sem fios (transmissores) em diversos espaços, desde empresas, universidades, espaços públicos, etc., o que coloca problemas relacionados com a optimização do número destes dispositivos, por questões económicas sobretudo. Muitos dos edifícios, onde este tipo de problema se coloca, podem ser representados por estruturas geométricas, como polígonos, para assim o problema poder ser mais facilmente estudado.

Assim, a pesquisa de um algoritmo para a determinação da região de cobertura de um dado k-transmissor num polígono torna-se uma parte importante na pesquisa de soluções para o problema referido. Uma vez que se pretende que este algoritmo seja útil na investigação de novas soluções para o problema, é importante ter em conta o seu desempenho, isto é, a sua complexidade, principalmente no que se refere ao tempo dispendido para encontrar uma solução correcta.

Uma vez que é conhecida a existência de um algoritmo ([5]) para o problema da determinação da região coberta por um k-transmissor, o objectivo é fazer uma pesquisa no sentido de saber se será possível melhorar o seu desempenho, já que esse algoritmo apresenta complexidade temporal de $O(n^2)$. Assim, pretende-se desenvolver novos algoritmos para a resolução deste problema, que apresentem uma maior eficiência e construir uma aplicação onde serão implementados esses algoritmos. Nesse sentido, um dos alvos deste trabalho será também o de testar e comparar o desempenho dos diferentes algoritmos, com recurso a uma aplicação desenvolvida anteriormente para obter polígonos simples aleatoriamente ([6]). Assim, a implementação feita ao longo do trabalho deverá também integrar esta última, de modo a que seja gerado um polígono aleatório, escolhido um ponto para fixar um k-transmissor e calculada a respectiva região de k-cobertura.

1.3 Organização do texto

Esta dissertação é constituída por 4 capítulos. Neste primeiro capítulo, é feita uma introdução ao assunto tratado ao longo do trabalho, sendo formalmente apresentado o problema da determinação da região de cobertura de um k-transmissor, referidas as motivações para o estudo desse problema e descrito resumidamente o trabalho realizado nesta área até ao momento por diversos autores. Ainda neste capítulo são introduzidos os principais conceitos que irão sendo utilizados ao longo da dissertação.

No Capítulo 2, são descritos e analisados os algoritmos existentes para a resolução do problema proposto e apresentadas novas abordagens para a resolução desse problema. Cada um dos 4 algoritmos apresentados é estudado em detalhe, analisando-se os vários

passos que executa e apresentado-se formalmente o algoritmo e será analisado do ponto de vista da sua complexidade temporal.

Segue-se o Capítulo 3, onde será descrita a aplicação onde se implementaram os algoritmos descritos no capítulo anterior. Será apresentado o contributo duma aplicação anterior [6], na geração aleatória de polígonos que permitirá depois a aplicação dos algoritmos de determinação da região de cobertura de um k-transmissor colocado num desses polígonos. Em seguida, descreve-se resumidamente a implementação feita. Serão também descritas as características principais da interface gráfica criada para a aplicação feita, por forma a esta ser mais facilmente utilizada por parte dos utilizadores. Por último, nesse capítulo, é efectuada uma análise dos resultados obtidos experimentalmente, utilizando a aplicação desenvolvida, e realizada uma comparação dos algoritmos estudados e implementados relativamente aos resultados produzidos.

Finalmente, no Capítulo 4, apresentam-se de forma sucinta os resultados mais relevantes e as principais conclusões a tirar deste trabalho, bem como alguns dos problemas em aberto relativamente a esta temática.

1.4 Estado de Arte do Problema

Nesta secção, são referidos os principais resultados existentes nesta área. Os mais relevantes para este trabalho serão descritos com mais detalhe, em secções seguintes.

Os problemas de visibilidade começaram a evidenciar-se com o denominado Problema da Galeria de Arte, colocado por Victor Klee em 1973 (de acordo com Honsberg [1]), que pretendia saber o número mínimo de guardas suficientes para cobrir o interior de uma sala de uma galeria de arte com n paredes. Impelido por este problema, Vasek Chvátal [2] mostrou, em 1975, que $\lfloor n/3 \rfloor$ guardas são ocasionalmente necessários e sempre suficientes para guardar um polígono simples com n vértices. Desde então, muitos investigadores têm estudado diversas variantes deste problema e diversas formas de visibilidade.

Para o conceito clássico de visibilidade, D. Lee [7] apresenta, em 1983, um algoritmo para calcular a região de visibilidade de $x \in P$, $V(x)$, em tempo linear, que será descrito no próximo capítulo. Em 1987, J. O'Rourke [8] publica o primeiro livro dedicado ao Problema da Galeria de Arte; além disso, muitos artigos têm sido escritos e resolvidos alguns dos importantes problemas colocados. Recentemente, em 2007, foi escrito, por Gosh [9], um livro sobre algoritmos de visibilidade em duas dimensões, entre os quais algoritmos para a construção de polígonos de visibilidade (como o algoritmo de Lee, descrito no próximo capítulo).

Mais recentemente, os aparelhos de rede wireless levaram à generalização do problema clássico de iluminação de polígonos, que se baseia em, em vez de se modelar por exemplo uma fonte de luz, modelar um dispositivo wireless cujo sinal consegue atravessar um dado número k de paredes. Isto levou à formulação de diversos novos problemas, como o de

saber o número destes dispositivos sempre suficientes, e sempre necessários, para cobrir um polígono P.

Assim, relativamente ao estudo em particular do problema de k-cobertura, motivado pelas comunicações de redes sem fio cujo sinal é enfraquecido por obstáculos, Aichholzer et al [4] publicaram, em 2009, um artigo com limites do número de k-transmissores necessários para cobrir polígonos monótonos e ortogonais monótonos, que representam a maioria dos edifícios na vida real, para $k > 0$. Na sequência da referência a este importante artigo, apresentam-se em seguida os principais resultados desse estudo.

No estudo que fizeram, os autores começam por provar que se consegue iluminar qualquer $(k+2)$-gon com um k-modem colocado em qualquer ponto do polígono.

Depois, conseguem provar se consegue iluminar qualquer n-gon monótono com $\lceil n/2k \rceil$ k-modems.

Afirmam também que, para $k = 1, 2, 3$, qualquer n-gon monótono pode consegue ser iluminado usando um número inferior de k-transmissores, $\lceil n/k + 4 \rceil$ (ver figura 1.8, extraída de [4]).

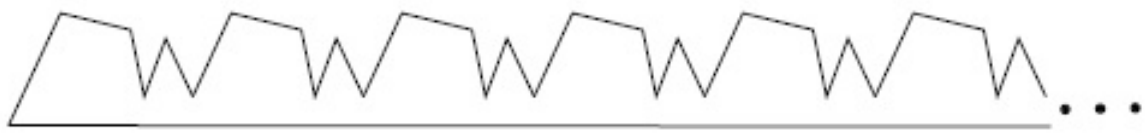


Figura 1.8: Um n-gon monótono que requer $\lceil n/5 \rceil$ 1-modems para ser iluminado.

No caso em que os polígonos monótonos são também ortogonais, os autores começam por provar que qualquer polígono P ortogonal com, no máximo, $k+4$ vértices é iluminado por um k-transmissor colocado em qualquer ponto de P e que para qualquer $(k+5)$-gon ortogonal x-monótono existe um ponto na sua aresta mais à esquerda (ou mais à direita) onde pode ser colocado um k-transmissor que ilumina o polígono. Mostram ainda que qualquer $(2k+6)$-gon ortogonal x-monótono pode ser iluminado com um k-transmissor. Por fim, mostram que:

Teorema 1.1. *Qualquer polígono ortogonal x-monótono com n vértices consegue ser iluminado com $\lceil n - 2/2k + 4 \rceil$ k-transmissores.*

Tal como no caso anterior, mostram um exemplo (ver figura 1.9a, extraída de [4]) de um destes polígonos que pode ser iluminado com $\lceil n - 2/2k + 4 \rceil$ k-transmissores.

Se k for par, a mesma figura (1.9a) é um exemplo se um polígono ortogonal monótono que pode ser iluminado com apenas $\lceil n - 2/2k + 6 \rceil$ k-transmissores. Contudo, para $k = 1$, a figura 1.9b (extraída de [4]) é um exemplo de um n-gon ortogonal monótono que requer

$\lceil n - 2/6 \rceil$ 1-transmissor para ser iluminado, caso que não se aplica o limite inferior anteriormente apresentado.

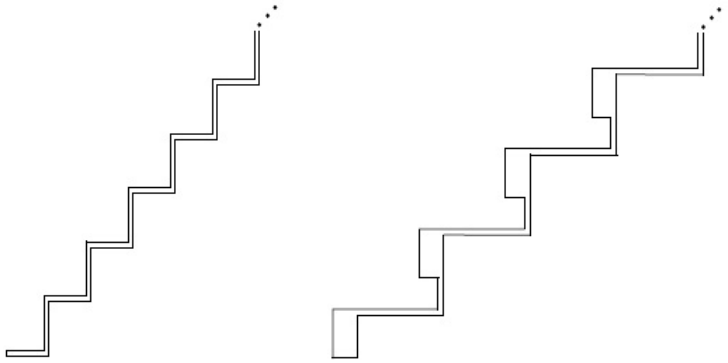


Figura 1.9: Polígonos ortogonais monótonos que exemplificam os limites enunciados.

Aparte de alguns erros em alguns limites (que por isso não foram aqui apresentados), os limites inferiores encontrados pelos autores para o número de k-transmissores necessários, são os melhores até agora conhecidos. Daí a importância destes resultados.

Contudo, o problema para polígonos arbitrários e ortogonais permaneceu em aberto. Ainda em 2009, Fabila-Monroy, Vargas e Urrutia [10] analisaram esta noção de visibilidade para outras configurações geométricas.

O problema clássico de vigiar um polígono simples é NP-difícil [11, 12] e acredita-se que também o seja o problema de encontrar o número mínimo de k-transmissores suficientes para cobrir um polígono simples [4]. Nesse sentido, Abellanas et al [13] aplicaram, em 2006, técnicas meta-heurísticas para resolver algumas variantes dos problemas de visibilidade. Uma meta-heurística é um método para resolver de forma genérica diferentes problemas de optimização, para os quais, geralmente, não se conhecem algoritmos eficientes. Seguindo esta estratégia, Martins [5] estudou formas de calcular o número mínimo de guardas colocados em vértices suficientes para cobrir um polígono. No estudo desenvolvido por Martins [5], foi proposto um algoritmo para construir a região coberta por um k-transmissor situado num ponto de um polígono simples de n vértices, para todos os valores possíveis de k. Este algoritmo será pormenorizadamente descrito no capítulo seguinte.

Existe, actualmente, a necessidade de um algoritmo eficazmente implementado, para prosseguir com estudos do problema do mínimo número de k-transmissores em polígonos simples.

1.5 Conceitos e notações

Seguidamente são apresentadas algumas definições e notações que serão utilizadas ao longo deste trabalho e que servirão para a compreensão geral do texto e de novos conceitos que será necessário introduzir, oportunamente, nos capítulos seguintes. Nas secções anteriores já foram referidos alguns conceitos, que agora serão também aqui expostos e clarificados.

Primeiramente, é importante esclarecer os vários tipos de visibilidade. Na definição geral, já apresentada, de visibilidade clássica admite-se, então, que pq passe por um vértice (côncavo) ou contenha uma aresta de P.

Não considerando a situação degenerada em que existem 3 vértices colineares, um vértice de P diz-se convexo se a amplitude do ângulo interno nele formado for inferior a π. Se a amplitude for superior a π, o vértice diz-se côncavo ou reflexo. (ver figura 1.10)

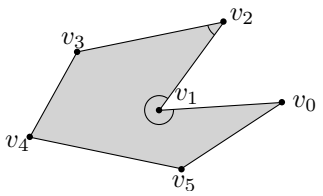


Figura 1.10: O vértice v_1 é côncavo e o vértice v_2 é convexo.

Além do conceito geral de visibilidade, outros tipos de visibilidade se podem considerar em relação a polígonos simples. Eis aquele que é importante reter, a par do conceito geral, para explicações futuras ao longo desta dissertação:

Definição 1.8. *Dois pontos p e q de um polígono P dizem-se claramente visíveis se pq estiver totalmente contido em $int(P)$ (ver figura 1.11).*

Note-se que, para este tipo de visibilidade, não é permitido que pq contenha pontos de $fr(P)$.

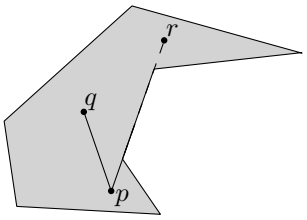


Figura 1.11: O ponto p vê claramente o ponto q, mas não o r.

Ao longo deste trabalho, quando não se especificar qual o tipo de visibilidade clássica assume-se que se trata do conceito geral de visibilidade.

Para esclarecermos algumas propriedades da região de cobertura (ou k-cobertura), é importante ter em conta as seguinte definições:

Definição 1.9. *Um polígono diz-se convexo se, para todo $x, y \in P$, o segmento de recta $xy \subset P$ (ver figura 1.12).*

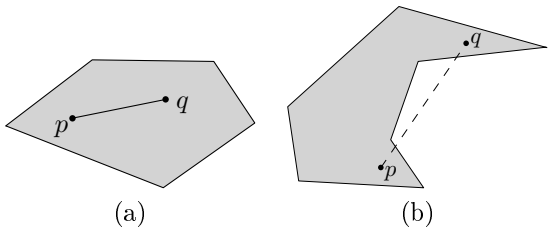


Figura 1.12: (a) Polígono convexo; (b) Polígono côncavo.

Definição 1.10. *O menor conjunto convexo que contém um polígono P designa-se invólucro convexo e denota-se por $CH(P)$ (do inglês, convex hull). (ver figura 1.13)*

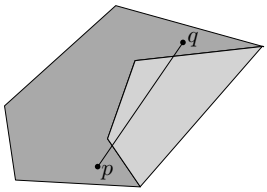


Figura 1.13: Invólucro convexo (toda a região sombreada) de um polígono P (apenas a região sombreada mais escura).

Assim, se um polígono simples P é convexo então o seu invólucro convexo coincide com o próprio polígono P.

Definição 1.11. *Um polígono simples P diz-se estrelado se existe pelo menos um ponto $q \in P$ tal que para todo o $p \in P, pq \subset P$ (ver figura 1.14).*

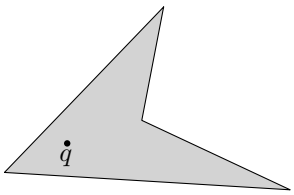


Figura 1.14: Polígono estrelado.

Definição 1.12. *Chama-se núcleo de um polígono P ao conjunto de pontos $K \subseteq P$, tal que todos os pontos de P são visíveis a partir de todos os pontos de K. Formalmente define-se da seguinte forma: $K = \{p \in P : \forall q \in P, p \text{ vê } q\} = \{p \in P : \forall q \in P, pq \cap P = pq\}$ (ver figura 1.15).*

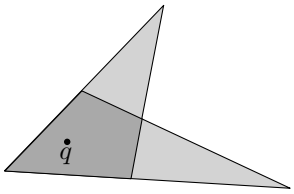


Figura 1.15: Núcleo de um polígono P.

Seguem-se alguns resultados associados ao núcleo de um polígono P:

1. $K(P)$ é um conjunto convexo;
2. $K(P) = P \Leftrightarrow P$ é convexo;
3. $K(P) = P \Leftrightarrow P$ é estrelado.

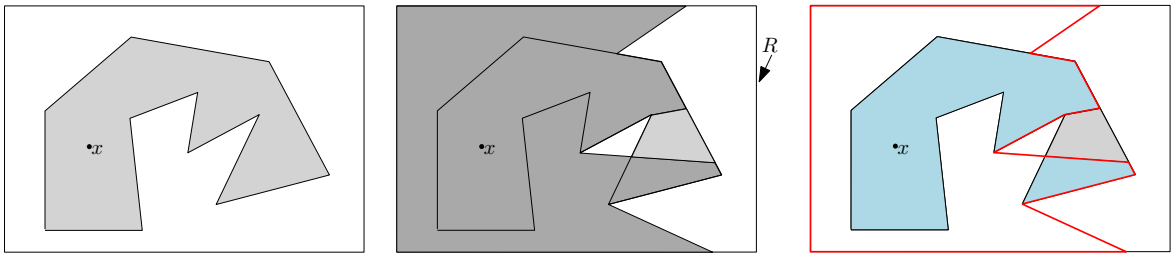
No caso da visibilidade clássica, atendendo à noção de polígono de visibilidade, pode-se agora dizer que qualquer polígono de visibilidade de um ponto $q \in P$, $V(q)$, é estrelado, pois $q \in K(V(q))$.

Já no caso caso da k-cobertura, $k > 0$, a região coberta por um k-transmissor colocado num ponto $x \in P$ pode ir mais além da fronteira do polígono P, pelo que, ao longo deste trabalho, se considerará que P está contido num rectângulo R e a região de k-cobertura será construída no interior de R. Mais concretamente, este procedimento é feito, sobretudo, por dois motivos: (i) para que a região de k-cobertura construída seja limitada, o que, de facto, corresponde à realidade pois o sinal de rede wireless tem um alcance limitado, e (ii) por razões práticas e de simplicidade de implementação.

Deste modo, a região coberta por x será sempre limitada e será denotada por $Vis_k(x, P)$.

Assim sendo, neste caso, e visto que a região de k-cobertura é sempre conexa, a região de k-cobertura também resultará num polígono estrelado.

Definição 1.13. *Uma região diz-se conexa se dois pontos quaisquer da região podem ser ligados por uma curva inteiramente dentro da região (ver figura 1.16).*



(a) (b) (c)

Figura 1.16: Conexidade da região de cobertura: (a) Polígono P, com 2-transmissor colocado em x; (b) Região de 2-cobertura (região mais escura - conexa); (c) Região de 2-cobertura restrita ao polígono P (regiões azuis - desconexa).

Como se vê na figura 1.16, a região de k-cobertura é sempre conexa, apesar de a mesma região restrita ao polígono poder não o ser. É a conexidade da região de k-cobertura que irá permitir adoptar determinadas estratégias nos algoritmos de determinação da região de k-cobertura de um k-transmissor.

Definição 1.14. *Um polígono é ortogonal se o conjunto das suas arestas é formado apenas por segmentos de recta horizontais e verticais. Deste modo os seus ângulos internos têm amplitude* $\frac{\pi}{2}$ *ou* $\frac{3\pi}{2}$*. (ver figura 1.17)*

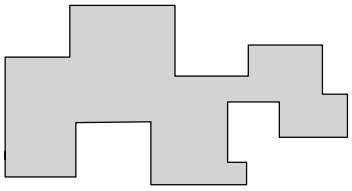


Figura 1.17: Polígono ortogonal.

Definição 1.15. *Um polígono P diz-se l-monótono, isto é, monótono em relação a um recta l, se a intersecção entre qualquer recta perpendicular a l e P é um ponto, um segmento ou o vazio. (ver exemplo da figura1.18)*

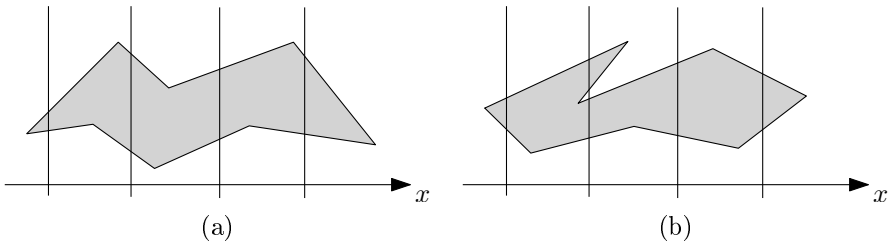


Figura 1.18: (a) Polígono x-monótono; (b) Polígono que não é x-monótono.

Na análise da eficiência dos algoritmos, falar-se-á em ordem de complexidade, noção que será agora explicitada.

Na comparação da eficiência dos diferentes algoritmos, é importante medir a quantidade de recursos utilizados na execução de cada um deles, nomeadamente ao nível do tempo que o algoritmo demora a encontrar uma solução e a nível de espaço de memória que necessita de ocupar. Geralmente, e tal como acontece nesta dissertação, a análise da eficiência restringe-se apenas à complexidade temporal, não havendo, em geral, problemas com a complexidade espacial. A noção de complexidade é medida usando a notação $O(g(n))$, onde n representa a quantidade de dados de entrado e $g(n)$ é a função tempo $T(n)$ (ou função espaço $S(n)$, no caso da complexidade espacial):

Definição 1.16. *Sejam f e g duas funções de domínio $\mathbb{N}$. Dizemos que a função f é $O(g(n))$ se, $\exists c \in \mathbb{R}^+, \exists N \in \mathbb{N} : |f(n)| \leq c|g(n)|, \forall n \geq N$. (ver figura 1.19)*

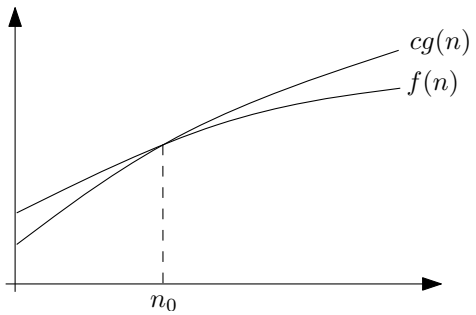


Figura 1.19: f tem ordem de complexidade $g(n)$.

Ao usar-se a notação O (ordem de complexidade), pode-se descrever o tempo de execução de um algoritmo, identificando as operações elementares e o máximo número de vezes que estas são executadas, isto é, inspeccionando simplesmente a estrutura do algoritmo para o pior caso.

Capítulo 2

Algoritmos para Determinar a Região de Cobertura

Conforme já foi referido, é objectivo desta tese encontrar algoritmos eficientes para a determinação da região de cobertura de um k-transmissor, dado um polígono P e a localização do transmissor, bem como o seu valor k. Até à data do início deste estudo, era conhecido apenas um algoritmo para a resolução deste problema com complexidade quadrática, $O(n^2)$, além do algoritmo de Lee, que permite encontrar o polígono de 0-visibilidade de um ponto num dado polígono (visibilidade clássica).

É sabido também que existe actualmente a necessidade de implementar um algoritmo que resolva de forma eficiente o referido problema, para prosseguir com estudos sobre o problema de optimizar o número de k-transmissores em polígonos simples. Assim, faz sentido realizar um estudo no sentido de encontrar outras formas de resolver o problema da determinação da região de cobertura de um k-transmissor de forma eficiente, isto é, encontrar outros algoritmos que, pelo menos, não tenham uma complexidade superior à do algoritmo já existente, $O(n^2)$.

Neste capítulo, será inicialmente apresentado o Algoritmo de Lee, principalmente pela sua importância histórica no estudo da visibilidade. Será também apresentado o referido algoritmo já existente, proposto por A. M. Martins [5], que será denominado de Algoritmo de Etiquetagem. De seguida, será apresentada uma secção onde se expõem os primeiros resultados do estudo feito para esta dissertação: com o intuito de melhorar a eficiência do algoritmo anterior, serão propostas melhorias ao algoritmo de Etiquetagem, construindo-se assim um novo algoritmo (baseado no primeiro), que será explicado nessa secção. Por último, será apresentado um outro novo algoritmo, com base na ideia de varrimento do polígono.

Para cada algoritmo apresentado, serão explicados todos os seus passos com a ilustração da aplicação de cada passo a um exemplo concreto. Será também analisada a complexidade temporal de cada algoritmo e, quando necessário, dadas algumas breves indicações sobre questões relacionadas com a sua implementação.

2.1 Algoritmo de Lee

O *algoritmo de Lee* foi apresentado por D. T. Lee [7], em 1983, e, formalmente, pretende resolver o seguinte problema de visibilidade clássica:

Dado um polígono simples P e um ponto q, encontrar o polígono de visibilidade $V(q)$.

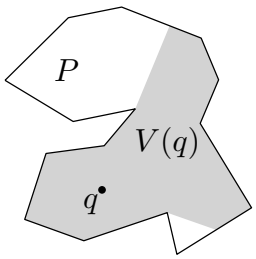


Figura 2.1: Polígono de visibilidade $V(q)$.

Inicialmente este problema foi considerado por Davis e Benedikt [14], que apresentaram um algoritmo de complexidade temporal quadrática, $\mathcal{O}(n^2)$. Mais tarde, Gindy e Avis [15] propuseram um algoritmo que resolve o problema em tempo linear, $\mathcal{O}(n)$, e, por último, Lee [7] apresentou um outro algoritmo com essa mesma complexidade e que é, actualmente, o mais conhecido. Por essa razão, apresenta-se agora o *algoritmo de Lee*.

2.1.1 Algumas considerações sobre o algoritmo

A apresentação de alguns conceitos, de seguida feita, será importante para a compreensão de alguns aspectos relativos a este algoritmo.

A fronteira de $V(q)$ é formada por arestas de $fr(P)$ cujos extremos são vértices visíveis a partir de q e outro tipo de arestas, às quais chamamos arestas construídas (ver figura 2.2).

Definição 2.1. *Uma aresta ab de $fr(V(q))$ diz-se uma aresta construída de $V(q)$ se:*
a) nenhum ponto de ab, excepto a e b, pertence a $fr(P)$;
b) os pontos q, a e b são colineares;
c) e a ou b é um vértice de P.

Observe-se que, uma vez conhecidas as arestas construídas de $V(q)$, é possível determinar a fronteira de $V(q)$ acrescentando $fr(P)$ entre duas arestas construídas consecutivas. Como cada aresta construída pode ser determinada em tempo linear, $O(n)$, um algoritmo simples deste género permite calcular $V(q)$ em tempo quadrático, $\mathcal{O}(n^2)$.

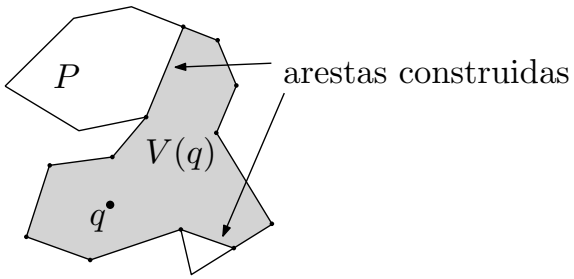


Figura 2.2: Arestas construídas de $V(q)$.

No entanto, como já foi referido, foram apresentados dois algoritmos com complexidade temporal mais reduzida, $O(n)$, para a resolução deste problema, o mais conhecido dos quais é o *algoritmo de Lee* [7], um algoritmo baseado no estudo de casos e que será aqui apresentado pela sua importância histórica e pela tentativa feita de generalizar este algoritmo ao caso da k-visibilidade. De referir que o *algoritmo de Lee* é o único algoritmo apresentado nesta dissertação que não foi implementado na aplicação desenvolvida, uma vez que não resolve o problema da k-cobertura abordado nesta tese (apenas determina a região de 0-visibilidade) e de, mesmo para $k = 0$, a sua aplicação falhar para alguns casos, como se explica seguidamente.

Assim, apesar da sua complexidade temporal mais reduzida que o primeiro algoritmo apresentado para a resolução do problema ([14]), é sabido que os dois algoritmos que apresentam complexidade $\mathcal{O}(n)$ (incluindo o *algoritmo de Lee*) podem falhar quando aplicados a polígonos ditos sinuosos, como mostrado por Joe e Simpson [16]. Antes de se apresentar a definição de polígono sinuoso, é necessário conhecer o conceito de número de revolução, a seguir apresentado.

Definição 2.2. *O número de revolução de um polígono simples P, em relação a um ponto q de P, é o número de revoluções (voltas completas) que $fr(P)$ faz em torno de q. (ver figura 2.3)*

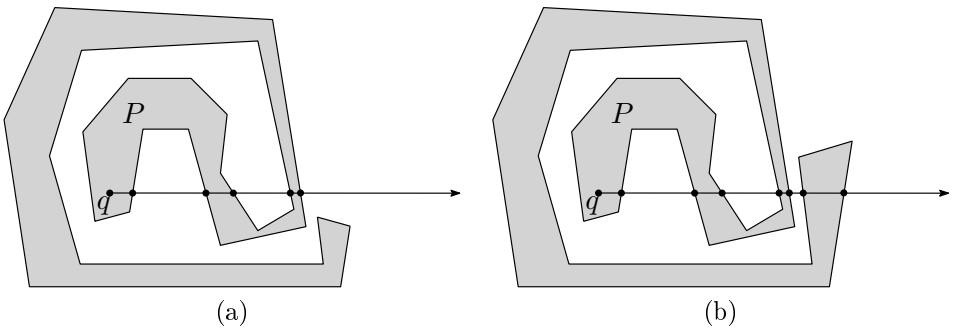


Figura 2.3: Número de revolução de P, relativamente a q, em (a) é 1 e em (b) é 2.

Definição 2.3. *Um polígono simples P diz-se não sinuoso se o número de revolução de P em relação a um ponto q de P for menor que dois (ver figura 2.3a). Caso contrário, diz-se sinuoso (ver figura 2.3b).*

Seguidamente, descreve-se e explica-se, passo a passo, o *algoritmo de Lee*, para a determinação do polígono de visibilidade $V(q)$ de um polígono simples não-sinuoso P com n vértices a partir de um ponto q, em tempo $O(n)$.

Relativamente à generalização do *algoritmo de Lee* ao caso da k-visibilidade, essa tentativa foi feita mas revelou ser demasiado complexa para se prosseguir com esse estudo pois, sendo o *algoritmo de Lee* um algoritmo baseado no estudo de casos, essa estratégia faria com que se tivesse de analisar inúmeros casos possíveis. Além disso, mesmo que tal algoritmo fosse desenvolvido para o caso da k-visibilidade, provavelmente, também não teria um resultado correcto para polígonos sinuosos, uma vez que seria baseado na ideia do *algoritmo de Lee*.

2.1.2 Descrição do algoritmo

O algoritmo conta com uma fase de pré-processamento, onde se verifica se o ponto q está no interior ou no exterior de P.

2.1.2.1 Pré-processamento

Passo 0: Verificar se q está no interior ou no exterior de P.

Se $q \in ext(P)$, constrói-se um polígono simples P' a partir de P tal que $q \in P'$ e $V(q) \subseteq P'$. Desta forma, o algoritmo utilizado para calcular $V(q)$, quando $q \in int(P)$, poderá ser utilizado para $q \in ext(P)$, isto é, para $q \in P'$ e $V(q) \subseteq P'$. Neste caso em que $q \in ext(P)$, é necessário verificar se q pertence ou não ao invólucro convexo de P, $CH(P)$, o que irá determinar a forma de construir P'. Como o invólucro convexo pode ser determinado em tempo $O(n)$, esta operação não vai alterar a complexidade do algoritmo que estamos a tratar.

Passo 0a: Se $q \notin CH(P)$, traçam-se duas semi-rectas, qv_i e qv_j, que partem de q e que são tangentes a $CH(P)$ em dois dos seus vértices, v_i e v_j. Verifica-se que todos os pontos de $fr(P)$ visíveis a partir de q estão entre v_i e v_j, em frente a q. Assim, para obter $fr(P')$, considera-se $fr(P)$ entre v_i e v_j e os segmentos de recta qv_i e qv_j.

Passo 0b: Se $q \in CH(P)$, considera-se $\overrightarrow{qv_k}$, o vector com origem em q que intersecta um qualquer vértice v_k de P. Considera-se também q', o primeiro ponto de intersecção de

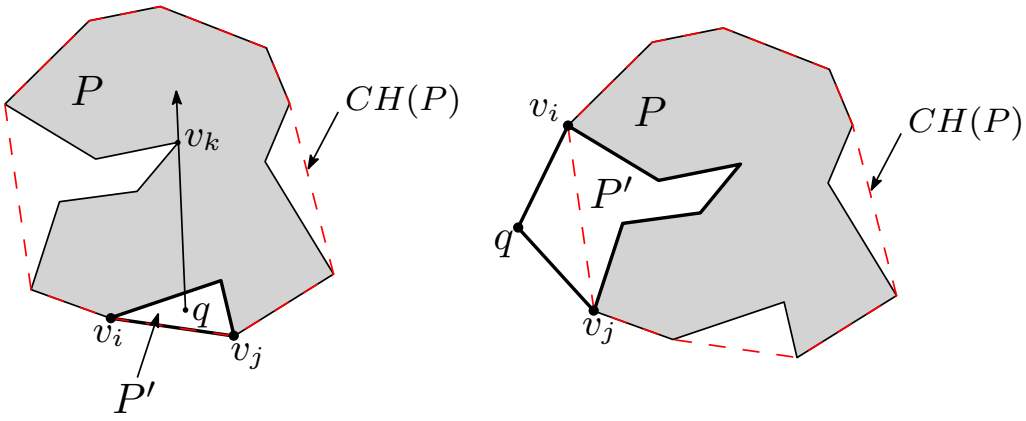


Figura 2.4: (a) $q \in CH(P)$;
(b) $q \notin CH(P)$.

$\overrightarrow{qv_k}$ com $fr(P)$. Partindo de q', percorre-se então $fr(P)$ no sentido negativo, até chegar a um vértice v_i de $CH(P)$. Repete-se o processo no sentido positivo, até chegar a outro vértice v_j de $CH(P)$. Observa-se que v_i e v_j são vértices consecutivos de $CH(P)$. Então, $fr(P')$ é a parte de $fr(P)$ entre esses dois vértices e a aresta de $CH(P)$ por eles formada.

Nos dois casos, obtém-se um novo polígono P' em que $q \in P'$, como se pretendia, para iniciar o algoritmo de Lee.

2.1.2.2 Determinação do polígono de visibilidade $V(q)$

De seguida, apresenta-se de forma explicativa o algoritmo referido. Convenciona-se que os vértices de P serão listado listados na ordem em que aparecem ao percorrer $fr(P)$ no sentido positivo.

Detonando por v_0 o ponto mais próximo de q de entre o vários pontos de intersecção de $fr(P)$ com a semi-recta horizontal com origem em q e à direita dele, coloca-se v_0 na pilha e inicializa-se $i = 1$. Nomeiam-se os vértices de P por $v_1, v_2, ..., v_n$, à medida que são atingidos quando se percorre $fr(P)$ no sentido positivo. Note-se que v_1 e v_n estão à esquerda e à direita de $\overrightarrow{qv_0}$, respectivamente.

O processo para determinar $V(q)$ inicia-se no vértice v_1 e percorre $fr(P)$ no sentido positivo até ao último vértice v_n. O algoritmo utiliza uma pilha, como estrutura de dados, onde coloca, pela ordem em que são encontrados, os vértice seleccionados e os extremos das arestas construídas de $fr(P)$ que são visíveis a partir de q. Assim, supondo que $fr(P)$ já foi analisada desde v_0 até ao vértice v_{i-1} e que v_i é o vértice que está a ser analisado, os

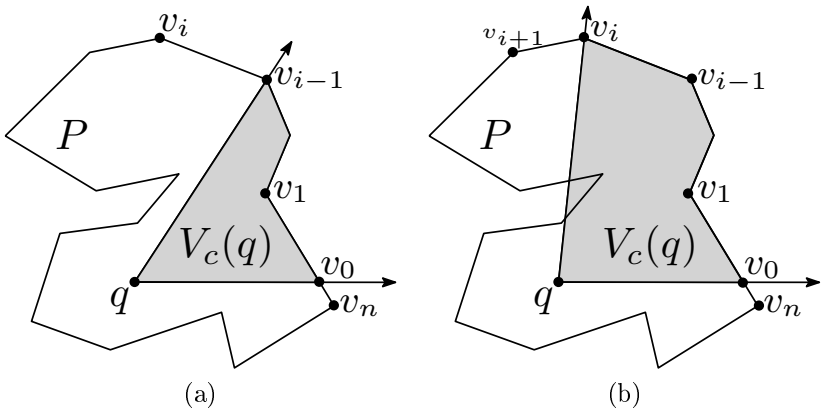


Figura 2.5: Região de visibilidade provisória: (a) $fr(v_0, v_{i-1})$ é visível a partir de q; (b) v_i é colocado no topo da pilha.

pontos de $fr(v_0, v_{i-1})$ (parte da fronteira entre v_0 e v_{i-1}, no sentido positivo) que verificam a condição de visibilidade até ao momento são acumulados na pilha, onde v_0 e v_{i-1} são a base e o topo da pilha, respectivamente. Desta forma, os pontos da pilha, juntamente com o ponto q, compõem um polígono constituído pelos vértices seleccionados até ao momento e por arestas construídas, polígono esse que forma uma região de visibilidade provisória, que se denota por $V_c(q)$ (em inglês, current visibility region) (ver figura 2.5).

Para a correcta construção do polígono de visibilidade, ao longo da execução do algoritmo é necessário ter em consideração a posição so vértice que está a ser analisado, o vértice v_i, em relação aos seus vértices adjacentes e ao ponto q, por forma a garantir que os pontos que são adicionados à pilha verificam a condição de visibilidade relativamente a q. Assim, os seguintes casos podem ocorrer:

Caso 1: o vértice v_i encontra-se à esquerda de $\overrightarrow{qv_{i-1}}$. (ver figura 2.5a)

Caso 2: o vértice v_i encontra-se à direita de $\overrightarrow{qv_{i-1}}$.
Caso 2a: o vértice v_i encontra-se à direita de $\overrightarrow{v_{i-2}v_{i-1}}$. (ver figura 2.6a)
Caso 2b: o vértice v_i encontra-se à esquerda de $\overrightarrow{v_{i-2}v_{i-1}}$. (ver figura 2.7b)

No caso 1, como os pontos entre v_{i-1} e v_i são visíveis a partir de q, adiciona-se o ponto v_i à pilha, como novo vértice de $V_c(q)$. O vértice v_{i+1} é, então, o próximo vértice a analisar.

No segundo caso, conclui-se que v_{i-1} e v_i não são ambos visíveis a partir de q. Isto é, ou $\overrightarrow{qv_i}$ intersecta $fr(v_0, v_{i-1})$ e, portanto, q não vê v_i (caso 2a), ou $\overrightarrow{qv_{i-1}}$ intersecta $fr(v_{i11}, v_n)$ e, consequentemente, q não vê v_{i-1} (caso 2b).

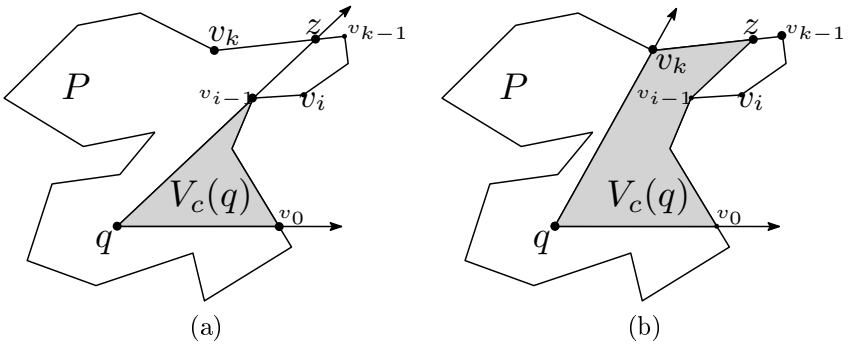


Figura 2.6: Caso2a: (a) os vértices de $fr(v_i, v_{k-1})$ não são visíveis a partir de q; (b) os pontos z e v_k são colocados na pilha e v_{k+1} é o novo v_i.

No caso 2a, note-se que pode suceder que, além de v_i, alguns vértices a seguir a este não sejam visíveis a partir de q. Nesta situação, o procedimento deve ser o seguinte:

Considere-se a primeira aresta $v_{k-1}v_k$ que resulta da intersecção entre $\overrightarrow{qv_{i-1}}$ e $fr(v_i, v_n)$ e seja z esse ponto de intersecção. Observa-se, então, que v_k está à esquerda de $\overrightarrow{qv_{i-1}}$ e que não há vértice de $fr(v_i, v_{k-1})$ visíveis a partir de q, pelo que z será o primeiro ponto, a seguir a v_{i-1}, visível a partir de q, sendo $v_{i-1}z$ uma aresta construída de $V(q)$; portanto, coloca-se o ponto z e v_k na pilha. Observe-se que o vértice q, v_{i-1} e z são colineares. Agora, v_{k+1} será o novo vértice v_i a analisar.

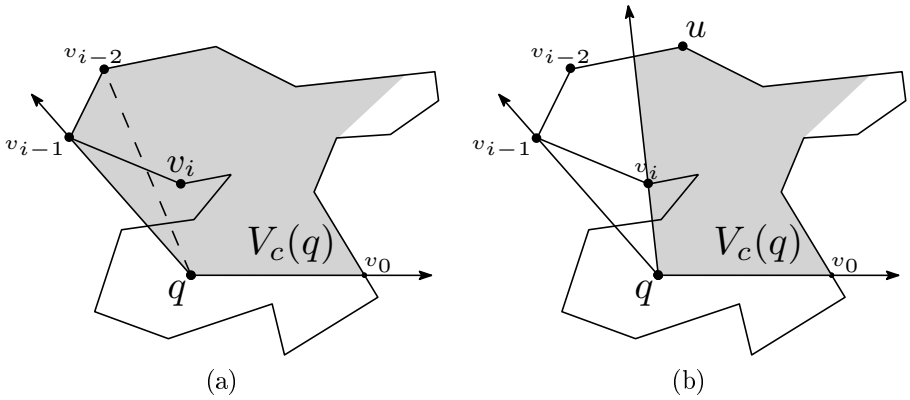


Figura 2.7: Caso2b: (a) a aresta $v_{i-1}v_i$ continua a intersectar $[uq]$; (b) a aresta $v_{i-1}v_i$ não intersecta $[uq]$.

Já no caso 2b, verifica-se que o vértice v_{i-1} não é visível a partir de q e que alguns dos vértices anteriores a esse poderão também não o ser. Note-se que, por esta altura, o vértice v_{i-1} encontra-se na pilha pois, estando este à esquerda do anterior, o passo anterior

foi executado recorrendo ao passo 1 do algoritmo. Portanto, retire-se agora o vértice v_{i-1} da pilha e denote-se por u o ponto que fica agora no topo da pilha. Agora, enquanto uq intersectar $v_{i-1}v_i$, vai-se retirando da pilha o vértice u; isto porque esses vértices que vão sendo eliminados da pilha não são visíveis a partir de q, dado que a sua visibilidade está a ser impedida pela aresta $v_{i-1}v_i$, a chamada *aresta da frente* (em inglês, *forward edge*). No final deste procedimento duas situações podem ocorrer: *i)* $v_{i-1}v_i$ não intersecta uq (ver figuras 2.7b e 2.8) ou *ii)* $v_{i-1}v_i$ intersecta uq (ver figura 2.9).

Na primeira situação (*i)*), observa-se a posição de v_{i+1} relativamente a $\overrightarrow{qv_i}$ para decidir se é efectuado ou não um novo retrocesso:

- se v_{i+1} estiver à direita de $\overrightarrow{qv_i}$ (ver figura 2.7b), o retrocesso continua com v_iv_{i+1} como nova aresta da frente;

- se, por outro lado, v_{i+1} estiver à esquerda de $\overrightarrow{qv_i}$ (ver figura 2.8), considere-se o ponto de intersecção m entre $\overrightarrow{qv_i}$ e a aresta do polígono que contém o ponto u, já identificado. Agora, há que verificar se v_{i+1} está à direita ou à esquerda de $\overrightarrow{v_{i-1}v_i}$:

    - no primeiro caso (ver figura 2.8a), simplesmente se adicionam os pontos m e v_i na pilha, verificando-se que v_im uma aresta construída de $V_c(q)$, e passa-se à análise do vértice seguinte a v_i, o vértice v_{i+1};
    - no segundo caso (ver figura 2.8b), percorre-se $fr(v_{i+1}, v_n)$ até encontrar um vértice v_k tal que a aresta $v_{k-1}v_k$ intersecta v_im e o retrocesso continua com $v_{k-1}v_k$ como nova aresta da frente e $v_i = v_k$ o vértice a analisar.

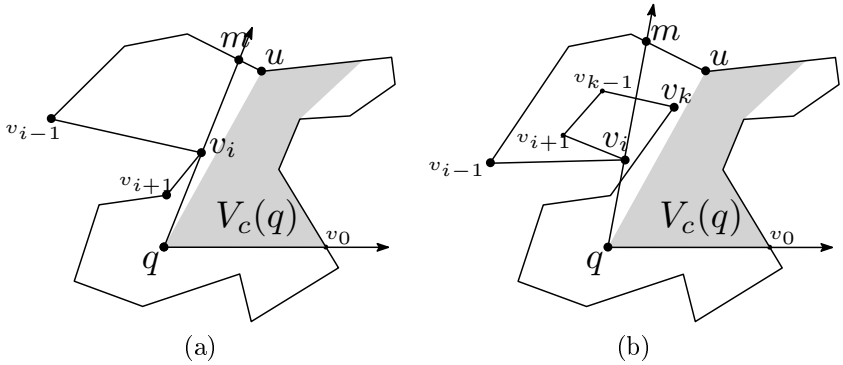


Figura 2.8: Caso2b *(i)*: (a) o retrocesso acaba e colocam-se os pontos m e v_i na pilha; (b) o retrocesso continua com $v_{k-1}v_k$ como nova aresta da frente.

Na segunda situação acima referida, (*ii)*), em que $v_{i-1}v_i$ intersecta uq, note-se que u não é vértice de P. Seja w o vértice logo abaixo de u na pilha e p o ponto de intersecção entre $v_{i-1}v_i$ e uq.

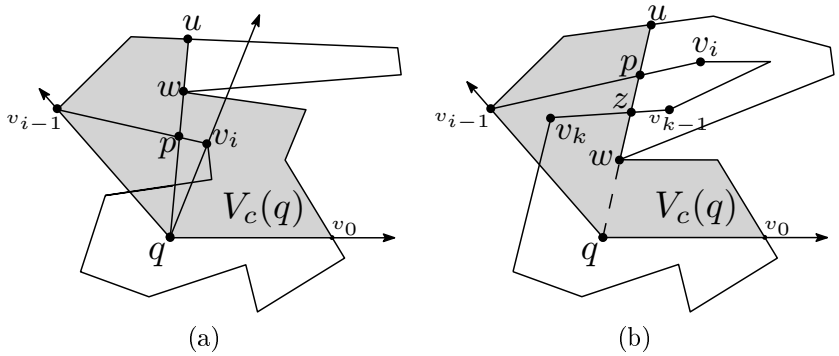


Figura 2.9: Caso2b *(ii)*: (a) a aresta $v_{i-1}v_i$ não intersecta uw; (b) a aresta $v_{i-1}v_i$ não intersecta uw em p e $v_{k-1}v_k$ intersecta wp num ponto z.

Se $p \in qw$ (ver figura 2.9a), então q não vê u nem w devido à aresta $v_{i-1}v_i$. Consequentemente, removem-se u e w da pilha (sendo u o ponto que agora fica no topo da pilha) e, conservando-se $v_{i-1}v_i$ como aresta da frente, continua-se o processo de retrocesso.

Caso contrário (ver figura 2.9b), então $p \in uw$. Para este caso, o procedimento consiste em percorrer $fr(v_i, v_n)$ até encontrar um vértice v_k tal que a aresta $v_{k-1}v_k$ intersecte wp num ponto z. Observe-se que todos os pontos de $fr(w, z)$, excepto w e z, não são visíveis a partir de q, pelo que se retiram esses pontos da pilha, mantendo-se w no topo, e acrescentam-se os pontos z e v_k na pilha. Agora, v_{k+1} será o novo vértice a analisar.

2.1.3 Análise da complexidade algorítmica

Esta secção aborda a complexidade temporal do *algoritmo de Lee*, de acordo o que foi descrito.

O algoritmo percorre a fronteira do polígono P e analisa a visibilidade de cada aresta de acordo com a sua posição, para a construção do polígono de visibilidade $V(q)$. Assim, uma vez que cada vértice é analisado uma só vez e apenas pode ser inserido uma vez na pilha, usada como estrutura de dados para armazenar os vértices de $V(q)$, e, se removido, não volta a entrar na pilha, nem a ser analisado, pode-se concluir que o *algoritmo de Lee* apresenta uma complexidade de $O(n)$, sendo n o número de vértice do polígono P.

No entanto, como foi referido, o algoritmo pode falhar quando aplicado a polígonos sinuosos.

2.1.4 Algoritmo formalizado

De seguida, apresenta-se o *algoritmo de Lee* resumido de maneira formal.

Algoritmo 1 Algoritmo de Lee

Entrada: Polígono P e ponto q.
Saída: Polígono de visibilidade $V(q)$.

Colocar v_0 na pilha e $i = 1$.

Passo 1. Colocar v_i no topo da pilha e fazer $i = i + 1$. Se $i = n + 1$, ir para o passo 8.

Passo 2. Se vi está à esquerda de $\overrightarrow{qv_{i-1}}$, ir para o passo 1.

Passo 3. Se vi está à direita de $\overrightarrow{qv_{i-1}}$ e de $\overrightarrow{v_{i-2}v_{i-1}}$, então

Passo 3a. Percorrer $fr(v_i, v_n)$ no sentido positivo, até encontrar um vértice v_k tal que $v_{k-1}v_k$ intersecte $\overrightarrow{qv_{i-1}}$. Seja z esse ponto de intersecção.

Passo 3b. Colocar z na pilha, fazer $i = k$ e ir para o passo 1.

Passo 4. Se está à direita de $\overrightarrow{qv_{i-1}}$ e à esquerda de $\overrightarrow{v_{i-2}v_{i-1}}$, então

Passo 4a. Retirar v_{i-1} da pilha. Seja u o ponto do topo da pilha.

Passo 4b. Enquanto $v_{i-1}v_i$ intersectar uq e u é um vértice de P, retira-se u da pilha.

Passo 5. Se $v_{i-1}v_i$ não intersecta uq, então

Passo 5a. Se v_{i+1} estiver à direita de $\overrightarrow{qv_i}$, fazer $i = i + 1$ e ir para o passo 4b.

Passo 5b. Seja m o ponto de intersecção entre $\overrightarrow{qv_i}$ e a aresta que contém u. Se v_{i+1} estiver também à direita de $\overrightarrow{v_{i-1}v_i}$, colocar m na pilha e ir para o passo 1.

Passo 5c. Percorrer $fr(v_{i+1}, v_n)$ até encontrar um vértice v_k tal que a aresta $v_{i-1}v_i$ intersecte v_im. Fazer $i = k$ e ir para o passo 4b.

Passo 6. Seja w o vértice imediatamente abaixo de u na pilha e p o ponto de intersecção entre $v_{i-1}v_i$ e uq. Se $p \in qw$, remover u e w da pilha e ir para o passo 4b.

Passo 7. Percorrer $fr(v_i, v_n)$ até encontrar um vértice v_k tal que a aresta $v_{k-1}v_k$ intersecte wp. Colocar o ponto de intersecção na pilha, fazer $i = k$ e ir para o passo 1.

Passo 8. Obter $V(q)$ retirando todos os vértices da pilha e parar.

2.2 Algoritmo de k-cobertura por Etiquetagem

Em 2009, foi proposto por A. M. Martins [5], o primeiro algoritmo para a determinação da região de cobertura de um k-transmissor. Este algoritmo surgiu da necessidade de responder ao seguinte problema:

Qual o número mínimo de k-modems em vértices necessários para cobrir um dado polígono P de n vértices, dado um valor de k?

Assumindo a possibilidade de que este problema seja $\mathcal{NP}$-difícil, a autora procurou usar métodos de solução aproximados e, dada a inexistência (até àquela data) de algoritmos que calculassem $Vis_k(x, P)$ para a necessária fase de pré-processamento antes da aplicação dos métodos referidos pela autora, foi proposto o algoritmo que a seguir se descreve. Existe uma pequena diferença entre o algoritmo descrito nesta dissertação e o mesmo algoritmo descrito pela autora [5], que será referida aquando da descrição do passo em que essa diferença se encontra, uma vez que a implementação do algoritmo foi feita de acordo com uma comunicação pessoal feita que vincula a alteração aqui apresentada. As imagens apresentadas são retiradas de [5].

Ao longo desta dissertação, o algoritmo a que esta secção se refere será chamado de *algoritmo de k-visibilidade por etiquetagem*, devido à estratégia seguida na sua execução de etiquetar segmentos da fronteira do polígono, que representam o número de obstáculos que o sinal emitido pelo k-transmissor terá de atravessar até chegar ao ponto em questão.

2.2.1 Preliminares do algoritmo

Seja P um polígono com n vértices, $V_P = v_0, v_1, ..., v_{n-1}$ e x um ponto de P onde está colocado um k-transmissor. A região de visibilidade construída por este algoritmo terá zonas do interior de P e zonas do seu exterior. Conforme já foi referido no capítulo anterior, para facilitar, considera-se que P está contido num rectângulo R e a região de k-cobertura é construída dentro de R.

Na execução do algoritmo, será necessário ter em conta á seguinte noção:

Definição 2.4. *Um vértice $v_i \in V_P$ é designado de vértice crítico relativamente a x se os vértices $v_{i-1} \in V_P$ e $v_{i+1} \in V_P$ estiverem no mesmo semi-plano definido pela recta xv_i. (ver figura 2.10)*

Estes vértices são chamados críticos porque influenciam determinadas mudanças no desenho da região de k-cobertura, dado que representam o aparecimento ou o fim de um

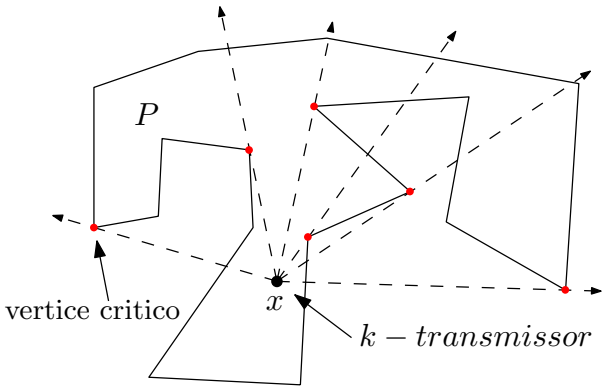


Figura 2.10: Vértices críticos (a vermelho) de um polígono P relativamente a x.

novo obstáculo (paredes) para o sinal do k-transmissor.

Por simplicidade, por vezes estes vértices serão chamados simplesmente vértices críticos, excluindo-se a referência do ponto em relação ao qual são vértices críticos, sempre que for claro que ponto de referência é esse.

Seguidamente é descrito o algoritmo de etiquetagem para a construção de $Vis_k(x, P)$, para todos os valores admissíveis de k, isto é, $0 \leq k \leq n$, conforme foi descrito pela primeira vez.

2.2.2 Descrição do algoritmo

Nesta primeira descrição, serão ignorados os seguintes casos degenerados:

- x é colinear com dois, ou mais, vértices críticos de P e
- x pertence a uma recta que passa sobre uma aresta de P.

Numa segunda fase, serão descritos os passos para o tratamento dos casos degenerados.

2.2.2.1 Descrição passo a passo

Os passos principais do algoritmo são os seguintes:

Passo 1: Encontrar os vértices críticos e numerá-los angularmente em torno de x. Desenhar todas as semi-rectas com origem em x passando pelos pontos críticos de P. Calcular, a partir dos pontos críticos, os pontos de intersecção q das semi-rectas com cada aresta de P (ver figura 2.11).

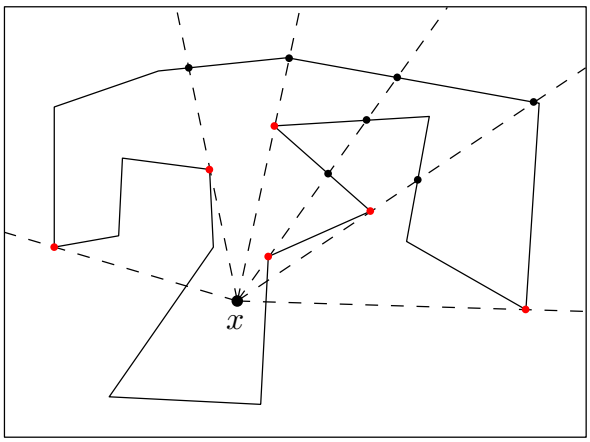
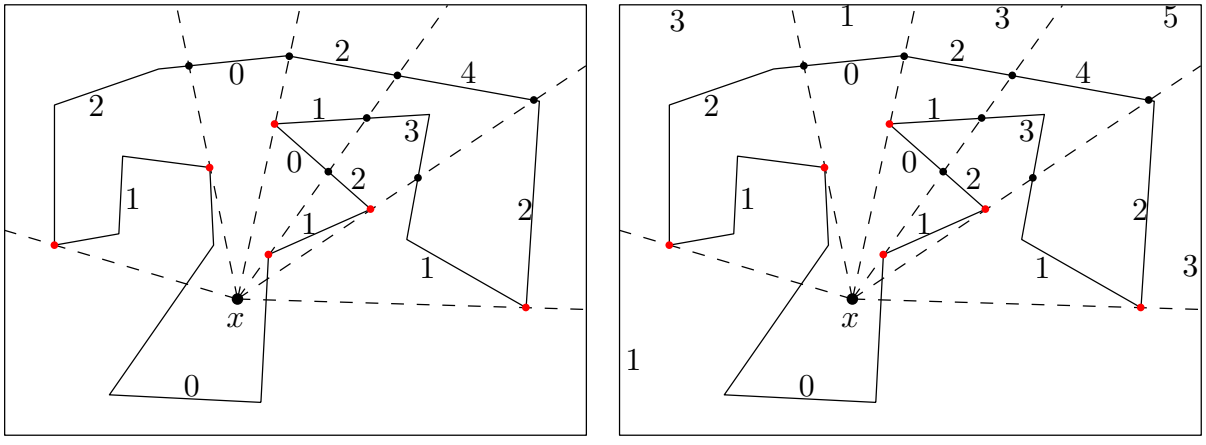


Figura 2.11: Semi-rectas e pontos de intersecção.

Passo 2: As semi-rectas dividem cada aresta do polígono em um ou mais segmentos. Etiquetar cada um desses segmentos com o número de arestas (paredes) atravessadas pela semi-recta, desde x até ao segmento em causa (ver figura 2.12a).

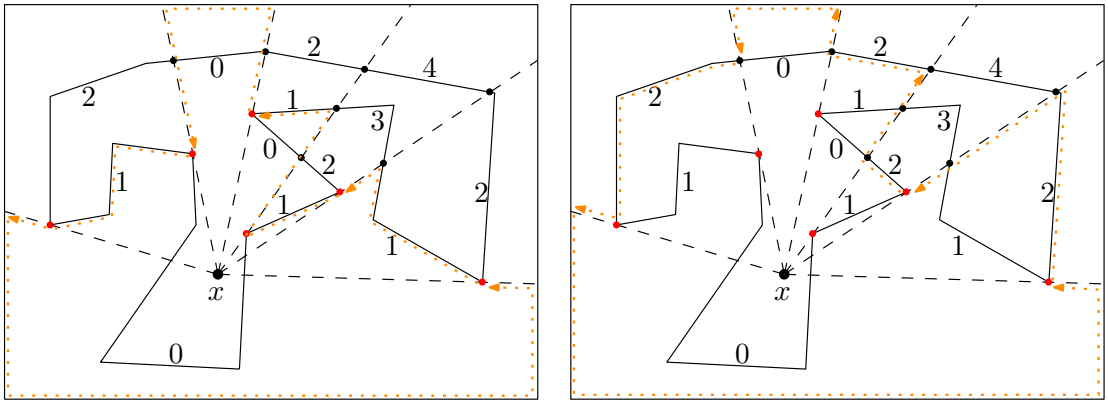
De acordo com a comunicação pessoal acima referida, este procedimento deve ser repetido para o rectângulo R que envolve o polígono, de modo a etiquetar também a sua fronteira (ver figura 2.12b). De facto, este passo não é necessário (razão pela qual é eliminado no algoritmo proposto com base neste, com intenção de melhorar a sua eficiência), mas foi realizado na implementação feita para a aplicação.



(a) (b)

Figura 2.12: Segmentos etiquetados: (a) Segmentos do polígono etiquetados; (b) Segmentos do polígono e do rectângulo etiquetados.

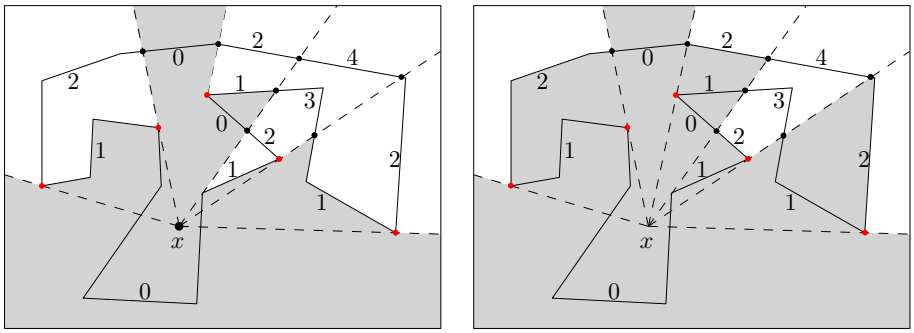
Passo 3: $Vis_k(x, P)$ é construído ligando, em ordem angular, todos os segmentos com a mesma etiqueta k, usando as semi-rectas incidentes. Para isso, consideramos um segmento $s1s2$, com uma etiqueta k, onde $s1 < s2$ em relação à ordem angular em torno de x. Assim, em $s1$ começa-se a desenhar a fronteira de $Vis_k(x, P)$, em direcção anti-horária. Depois, avança-se pela fronteira de P (se k é par avança-se em direcção anti-horária (ver exemplo da figura 2.13a); caso contrário, avança-se com direcção horária (ver exemplo da figura 2.13b)), até ser alcançada a intersecção de uma das semi-rectas com um segmento com uma etiqueta diferente k'. Se $k' > k$, terá de se percorrer a semi-recta (com origem no ponto x), em direcção a x, até ser cruzado um segmento com etiqueta k. Caso contrário, isto é, se $k' < k$, percorre-se a semi-recta encontrada no sentido oposto. Note-se que, se o segmento com etiqueta k não existir, deve-se percorrer a margem da caixa rectangular. Este procedimento continua até $Vis_k(x, P)$ ficar fechado (ver figura 2.14).



(a) (b)

Figura 2.13: Construção de: (a) $Vis_1(x, P)$; (b) $Vis_2(x, P)$.

As figuras seguintes (2.14a e 2.14b) mostram a regiões de cobertura $Vis_1(x, P)$ e $Vis_2(x, P)$.



(a) (b)

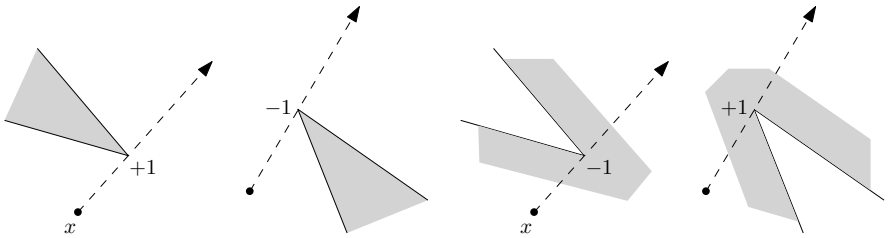
Figura 2.14: (a) $Vis_1(x, P)$; (b) $Vis_2(x, P)$.

O passo 2 será, agora, mais detalhadamente explicado.

1. Etiquetar cada vértice crítico com "+1" ou "−1". Sendo vi um vértice crítico, existem quatro regras a seguir para etiquetá-lo:

   1.1 Se $v_{i-1}v_iv_{i+1}$ faz uma curva para a esquerda e:
   (i) v_{i-1} encontra-se no lado positivo do vector $\overrightarrow{xv_i}$, então etiquetar v_i com "+1" (**regra 1**);
   (ii) v_{i-1} encontra-se no lado negativo do vector $\overrightarrow{xv_i}$, então etiquetar v_i com "−1" (**regra 2**);

   1.2 Se $v_{i-1}v_iv_{i+1}$ faz uma curva para a direita e:
   (i) v_{i-1} encontra-se no lado positivo do vector $\overrightarrow{xv_i}$, então etiquetar v_i com "−1" (**regra 3**);
   (ii) v_{i-1} encontra-se no lado negativo do vector $\overrightarrow{xv_i}$, então etiquetar v_i com "+1" (**regra 4**);



(a) Regra 1 (b) Regra 2 (c) Regra 3 (d) Regra 4

Figura 2.15: Regras para etiquetar os vértices críticos (as zonas sombreadas representam $int(P)$).

Estas etiquetas servirão para, posteriormente, ajudar na correcta etiquetagem dos segmentos ao longo da fronteira de P e representam um novo obstáculo que surge (etiquetas "+1"), percorrendo a fronteira em sentido positivo, ou o fim dum obstáculo (etiquetas "−1"). Assim, ao percorrer a fronteira em sentido positivo será possível, dado um valor dum determinado segmento, etiquetar o segmento seguinte com base na etiqueta destes vértices críticos e nas etiquetas dos pontos de intersecção (que a seguir se explicam).

Na figura 2.16 mostram-se os vértices críticos do polígono etiquetados, segundo as regras apresentadas.

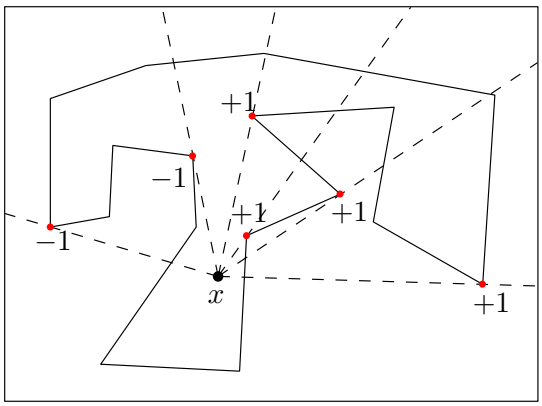


Figura 2.16: Polígono com vértices críticos etiquetados.

2. Etiquetar cada ponto de intersecção q identificado no passo 1 com "+2" ou "−2":

   Seja $q \in v_{j-1}v_j$, onde q pode ser igual a v_j. Se v_{j-1} e v_{i-1} se encontram do mesmo lado relativamente ao vector $\overrightarrow{xv_i}$, então etiquetar q com "−2´´; senão etiquetar q com "+2".

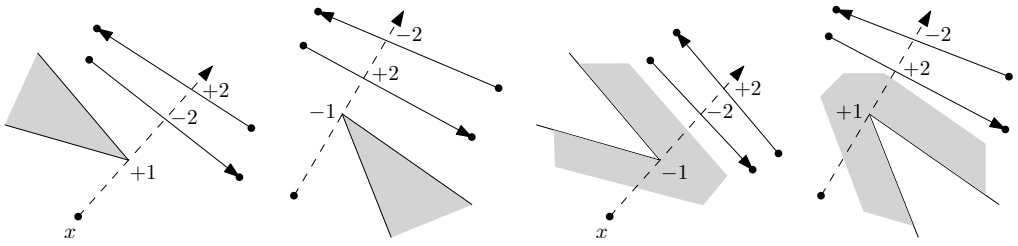


Figura 2.17: Regras para etiquetar os pontos de intersecção.

Da mesma forma que no caso dos vértices críticos, as etiquetas destes pontos representam (percorrendo a fronteira de P no sentido positivo) o aparecimento ("+2") ou o fim ("−2") de um obstáculo, obstáculos esses que têm a ver com o vértice crítico com o qual está relacionado o ponto de intersecção que se está a etiquetar. Neste caso, a etiqueta será "+2" ou "−2", pois o vértice crítico em questão representa dois obstáculos que são as duas paredes incidentes nesse vértice (ver figura 2.18).

3. Desenhar o vector horizontal com origem em x (para a direita de x) e detectar o primeiro ponto de intersecção z com $fr(P)$. Etiquetar com "0" o segmento ao qual z pertence (de z até ao vértice/ponto seguinte com uma etiqueta). Avançando pela fronteira de P até seguinte ao vértice/ponto etiquetado p, etiquetar os segmentos com a soma da etiqueta de p com a do segmento anterior. Executar este procedimento, até alcançar z novamente (ver figura 2.19).

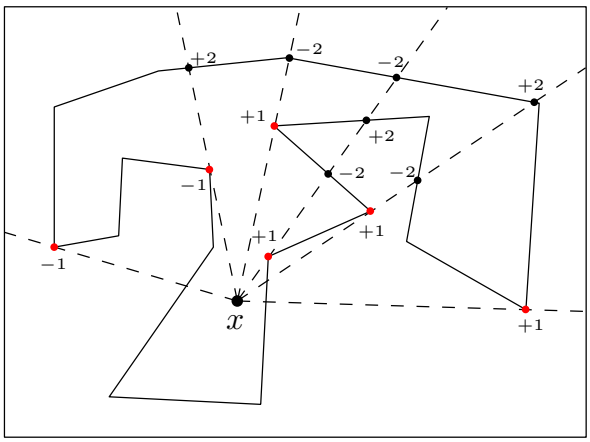
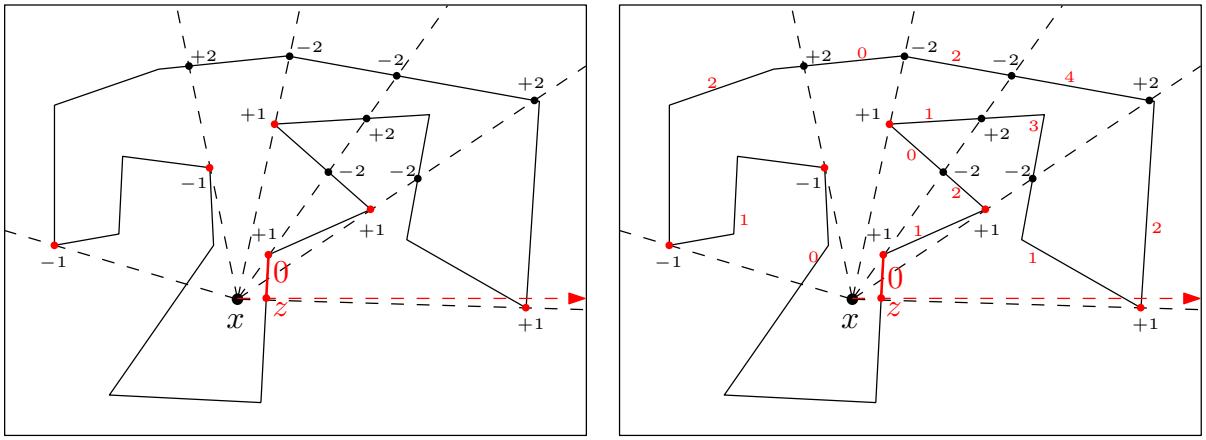


Figura 2.18: Pontos de intersecção etiquetados.



(a) Primeiro segmento etiquetado.

(b) Segmentos etiquetados.

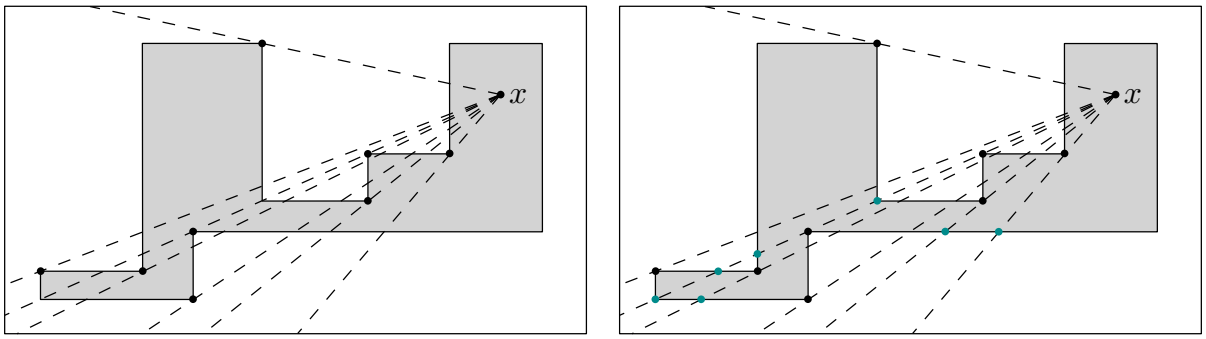
Figura 2.19: Processo de etiquetagem.

2.2.2.2 Tratamento de casos degenerados

Na descrição anterior do algoritmo, assumia-se que não se verificavam os seguintes casos degenerados:

- x é colinear com dois, ou mais, vértices críticos de P e
- x pertence a uma recta que passa sobre uma aresta de P.

Nesta secção, será feita uma adaptação do algoritmo ao caso degenerado em que x é colinear com dois ou mais vértices, de acordo com [5]. Continua-se a assumir que x não pertence a nenhuma recta que passa sobre uma aresta de P; isto porque, de acordo com a definição 1.6, este caso não é contemplado.



(a) (b)

Figura 2.20: Caso degenerado: (a) Uma das semi-recta tem 2 vértices críticos; (b) Pontos de intersecção.

As diferenças no algoritmo residem nos passos 1 e 2, os restantes passos permanecem iguais.

No algoritmo anteriormente descrito havia apenas um vértice crítico em cada semi-recta. Agora, cada semi-recta tem pelo menos um vértice crítico (ver figura 2.20a). Assim, no passo 1, os vértices críticos pertencentes a uma mesma semi-recta deverão ser ordenados, de acordo com a sua distância ao ponto x que representa o transmissor, do mais próximo para o mais afastado. Depois, a partir do vértice crítico mais próximo do transmissor, devem ser encontrados os pontos de intersecção de cada semi-recta com as arestas de P (ver figura 2.20b).

Relativamente ao passo 2, a etiquetagem dos vértices críticos é feita do mesmo modo. A diferença está na etiquetagem dos pontos de intersecção p_j (que não sejam vértices críticos) identificados no passo anterior, com "$+n$" ou "$-n$" ($n \in \mathbb{N}_0$). Agora, essa etiquetagem é feita do seguinte modo:

1. Inicializar n a 0.

2. Para cada vértice crítico v_i mais próximo de x do que p_j:

(a) Se p_j é um ponto extremo de uma aresta de P (isto é, se $p_j = v_j$, para algum $v_j \in V_P$), então:
i) $n = n + (-2)$, se v_{j-1} e v_{i-1} se encontram do mesmo lado de $\overrightarrow{xv_i}$;
ii) caso contrário, $n = n + (+2)$.

(b) Se p_j é um ponto não extremo de uma aresta de (isto é, se $p_j \in v_k v_{k+1}$, $p_j \neq v_k, v_{k+1}$, para algum $v_k \in V_P$) P, então:
i) $n = n + (-2)$, se v_j e v_{i-1} se encontram do mesmo lado de $\overrightarrow{xv_i}$;
ii) caso contrário, $n = n + (+2)$.

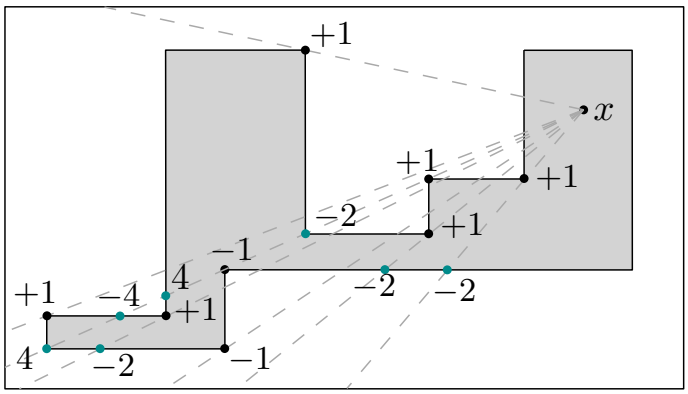
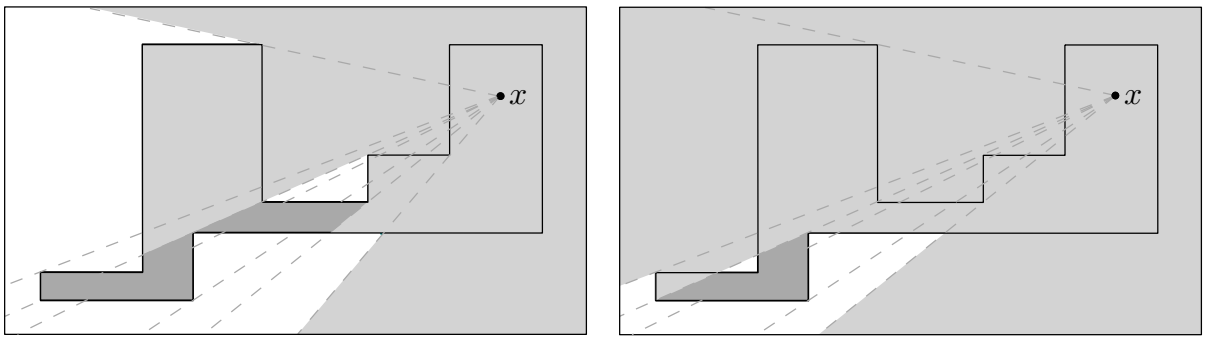


Figura 2.21: Caso degenerado: Pontos etiquetados.

A análise deste tipo de casos degenerados revela-se especialmente importante no caso de polígonos ortogonais, onde há maior probabilidade de haver vários vértices críticos colineares com o transmissor colocado dentro do polígono.

No entanto, na implementação deste algoritmo, não foram tratados os casos degenerados, por razões de tempo, de complexidade acrescentada que isso traria na implementação e, principalmente, porque os polígonos gerados pela aplicação para a execução deste algoritmo são polígonos aleatórios, o que significa que a probabilidade de um caso degenerado deste tipo acontecer é muito reduzida. O tratamento dos casos degenerados ao nível da aplicação desenvolvida é uma questão a tratar no futuro.



(a) (b)

Figura 2.22: Região de cobertura de um k-transmissor num polígono ortogonal de 14 vértices: (a) k=2; (b) k=4.

2.2.3 Algoritmo formalizado

Seguidamente, apresenta-se o *algoritmo de etiquetagem* resumido de maneira formal.

Algoritmo 2 Algoritmo de k-cobertura por Etiquetagem

Entrada: Polígono P, ponto x e valor k.
Saída: Região de visibilidade $Vis_k(x, P)$.

Passo 1. Construção das semi-rectas:

Passo 1a. Encontrar os vértices críticos em relação a x.

Passo 1b. Ordenar os vértices críticos angularmente em torno de x.

Passo 1c. Para cada vértice crítico encontrar os pontos de intersecção da semi-recta (com origem em x e passando pelo vértice crítico em questão) com cada aresta de P.

Passo 2. Etiquetagem:

Passo 2a. Etiquetar cada vértice crítico.

Passo 2b. Etiquetar cada ponto de intersecção q.

Passo 2c. Etiquetar segmentos.

Passo 3. Construção da região de cobertura $Vis_k(x, P)$.
Ligar, em ordem angular, todos os segmentos com a mesma etiqueta k, usando as semi-rectas incidentes.

2.2.4 Análise da complexidade algorítmica

Na análise da complexidade temporal do *algoritmo de etiquetagem* descrito, considera-se mais uma vez que n é o número de vértices do polígono P.

O passo 1, de encontrar os vértices críticos e, posteriormente, ordená-los angularmente em torno do transmissor, é feito em tempo $\mathcal{O}(nlogn)$, devido á ordenação angular.

Uma vez que cada semi-recta do passo 1 pode intersectar todas as n arestas do polígono e, do ponto de vista assimptótico, poderão existir n vértices críticos, logo n semi-rectas, o número de operações no passo 2 é quadrático, pois para cada semi-recta, há que percorrer o polígono em busca de pontos de intersecção.

Como tal, o número total de etiquetas dos segmentos, no passo 2, também é quadrático. Assim, o processo de etiquetagem decorre em tempo $\mathcal{O}(n^2)$.

A construção do polígono de visibilidade para cada valor fixado de k, passo 3, é feita em tempo linear, pois cada lado do polígono intervém um número constante de vezes.

Assim, pode-se concluir que a construção de $Vis_k(x, P)$ por este algoritmo é feita em tempo $O(n^2)$.

Este *algoritmo de k-cobertura por etiquetagem*, foi o primeiro a ser apresentado para a resolução do problema da determinação da região de cobertura de um k-transmissor. Como tal, alguns dos seus passos podem ser melhorados e/ou condensados num só, de forma a ser executados de maneira optimizada e mais eficiente.

Assim, o próximo algoritmo a ser apresentado, é um novo algoritmo, proposto com base neste algoritmo de etiquetagem. Deste ponto de vista, achou-se que seria interessante a implementação de ambos na aplicação feita, de forma a comparar os seus tempo de execução, para perceber até que ponto as alterações propostas implicam uma melhoria significativa na sua execução, como será explicado em capítulos seguintes.

2.3 Algoritmo de k-cobertura por Etiquetagem Melhorado

O algoritmo a seguir descrito baseia-se no algoritmo já existente, descrito na secção anterior, o Algoritmo de k-cobertura por Etiquetagem. A ideia geral continua a ser etiquetar os segmentos da linha poligonal, para depois formar a região de k-cobertura pretendida; no entanto, a maioria dos passos é feita de forma optimizada e eficiente com o objectivo de reduzir o número de operações a efectuar. Por esse motivo, ao longo desta dissertação, o algoritmo a que esta secção se refere será chamado de Algoritmo de k-cobertura por Etiquetagem Melhorado.

A ideia de melhorar o referido algoritmo surgiu perante a dificuldade sentida em desenvolver outros algoritmos baseados em novas estratégias que conduzissem a uma complexidade inferior a $O(n^2)$, complexidade presente no algoritmo já existente, o Algoritmo de Etiquetagem. Assim, o algoritmo que será descrito de seguida, apesar de ter uma complexidade assimptótica de $O(n^2)$, tal como o anterior, apresenta os passos de forma optimizada de forma que o número de operações em alguns casos é reduzido significativamente, o que poderá fazer com que seja mais aconselhável executar este novo algoritmo, preterindo o anterior.

Desta forma a descrição que se segue não contem alguns pormenores já descritos no algoritmo anterior, como as regras de etiquetagem, uma vez que esse processo é idêntico, concentrando-se antes nas principais mudanças em relação ao algoritmos anterior, que serão descritas detalhadamente. Na descrição feita de seguida, será realizada também a análise da complexidade, passo por passo, por forma a facilitar a comparação com o algoritmo anteriormente descrito.

2.3.1 Descrição geral do algoritmo e análise da sua complexidade

Seja P um polígono com n vértices $V_P = v_0, v_1, ..., v_{n-1}$, ordenados em sentido positivo, e x um ponto do interior de P onde está colocado um k-transmissor. Pretende-se construir a região coberta pelo k-transmissor, desde o ponto x, atravessando no máximo k paredes (arestas). P está contido num rectângulo R e a região de visibilidade será construída no interior de R.

Além disso, supõe-se que x está em posição geral, ou seja, não se consideram os seguintes casos degenerados: *i)* x é colinear com dois ou mais vértices críticos de P e *ii)* x pertence a uma recta que passa sobre uma aresta de P.

Tal como anteriormente, também será necessário ter presente a noção de vértice crítico (ver figura 2.10).

Os passos do algoritmo são:

Passo 1: Encontrar vértices críticos e etiquetá-los.

Este passo consiste em encontrar os vértices críticos de P etiquetando-os, segundo mesmas regras do algoritmo anterior, no momento em que cada um deles é encontrado.

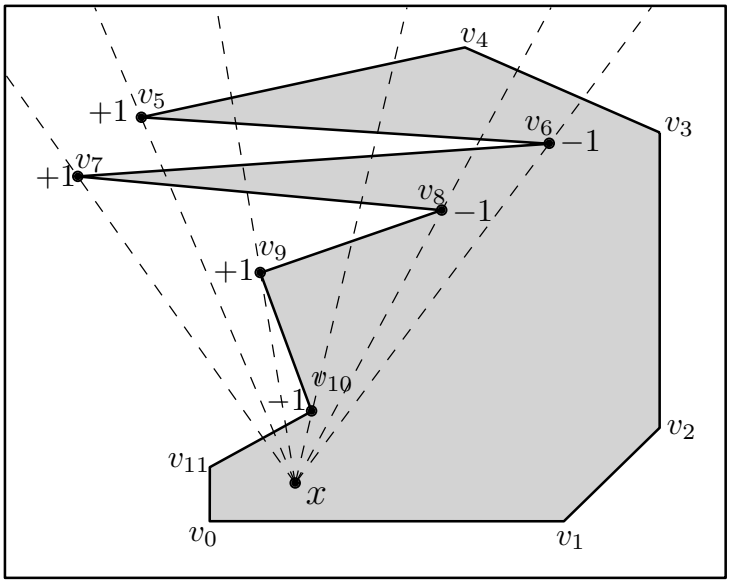


Figura 2.23: Passo 1: Vértices críticos etiquetados.

Deste modo, este passo permite condensar dois passos do anterior algoritmo num só; isto é, enquanto que no algoritmo anterior eram executados dois ciclo separados que percorriam os vértices do polígono (o primeiro ciclo à procura dos vértices críticos e o segundo etiquetando os vértices identificados), este algoritmo permite poupar um ciclo pois as duas operações são feitas num ciclo só. Deste modo, ao percorrer-se o polígono à procura dos

seus vértices críticos (analisando, para isso, em cada caso, a posição dos vértices anterior e seguinte), de cada vez que um destes vértices é encontrado, este é imediatamente etiquetado, de acordo com a informação do vértice anterior e seguinte.

Logo, este passo permite optimizar a forma como as operações são feitas em relação ao algoritmo anterior (apesar de, do ponto de vista assimptótico, este passo continuar a ter complexidade $O(n)$).

Passo 2: Ordenar dos vértices críticos angularmente em torno de x.

Este passo é feito tal como a ordenação feita para o algoritmos anteriormente descrito.

Para distinguir, a numeração dos vértices de P é feita denotando os vértices por v_i e os vértices críticos denotam-se aqui por c_j (ver figura 2.24).

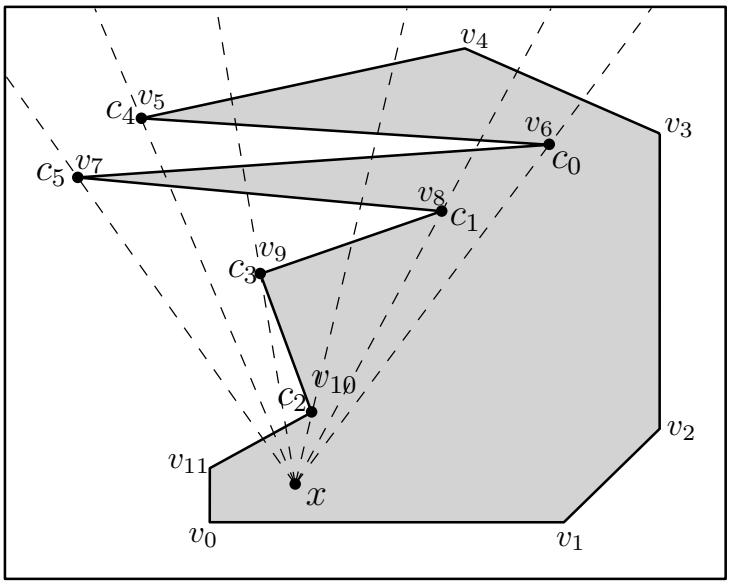


Figura 2.24: Passo 2: Vértices críticos ordenados.

Este passo vai ser importante para identificar, no passo seguinte, os pontos de intersecção p das semi-rectas (com origem em x e que passam por cada um dos vértices críticos) com a fronteira do polígono. Devido à ordenação angular dos vértices, este passo é feito em tempo $O(nlogn)$.

Passo 3: Encontrar os pontos de intersecção q das semi-rectas com o polígono e etiquetá-los.

Percorrendo cada aresta do polígono, deve-se encontrar as referidas intersecções de forma eficiente (a seguir descrita) e etiquetá-las de imediato, de acordo com as regras

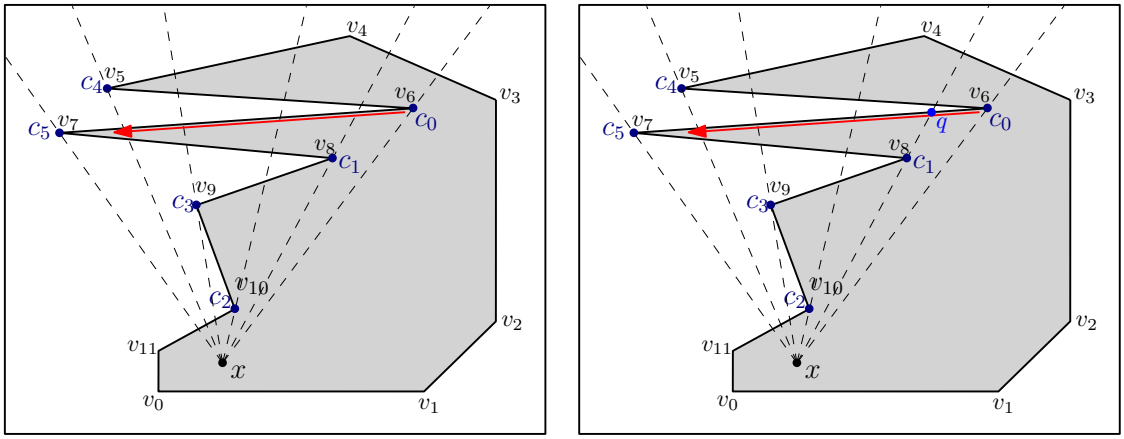
descritas no algoritmo anterior (ver figura 2.26a; note-se que os pontos de intersecção detectados se encontram já etiquetados). Da mesma forma que no primeiro passo descrito, isto vai permitir poupar um ciclo, uma vez que os pontos de intersecção são também etiquetados assim que encontrados.

A forma de pesquisa dos pontos de intersecção é a principal diferença em relação ao algoritmo anterior. Uma vez que este passo é executado com complexidade quadrática no algoritmo anterior, houve o cuidado de seguir uma estratégia que permitisse reduzir bastante o número de operações, apesar de, no caso assimptótico não haver diferença.

Na procura das intersecções de forma eficiente, recorre-se à ordenação feita no passo anterior e vão-se percorrendo as arestas de P:

Seja $c_j, j = 0$ o primeiro vértice crítico, que corresponde ao vértice $v_i \in V_P, i \in [0, n-1]$.

Começa-se, agora, a percorrer a fronteira de P (no sentido positivo) a partir de c_0. Deste modo, a aresta que está a ser percorrida denota-se por v_iv_{i+1} (ver figura 2.25a). Note-se que v_{i+1} pode ser ou não um vértice crítico.



(a) (b)

Figura 2.25: Passo 3: (a) primeira aresta de P percorrida; (b) primeiro ponto q encontrado (caso *i)*).

Denote-se por q o ponto de intersecção (se existir esse ponto) da semi-recta com origem em x, passando por um determinado vértice crítico, com a aresta que se está a analisar.

Ao percorrer a aresta v_iv_{i+1} (sempre no sentido positivo), procura-se a primeira intersecção nessa aresta (depois a segunda, se existir, e assim por diante), correspondente à semi-recta com origem em x e que, dependendo da orientação da aresta, passa:
i) pelo vértice crítico c_{j+1}, caso x esteja do lado esquerdo de v_iv_{i+1} (ver figuras 2.25b), ou

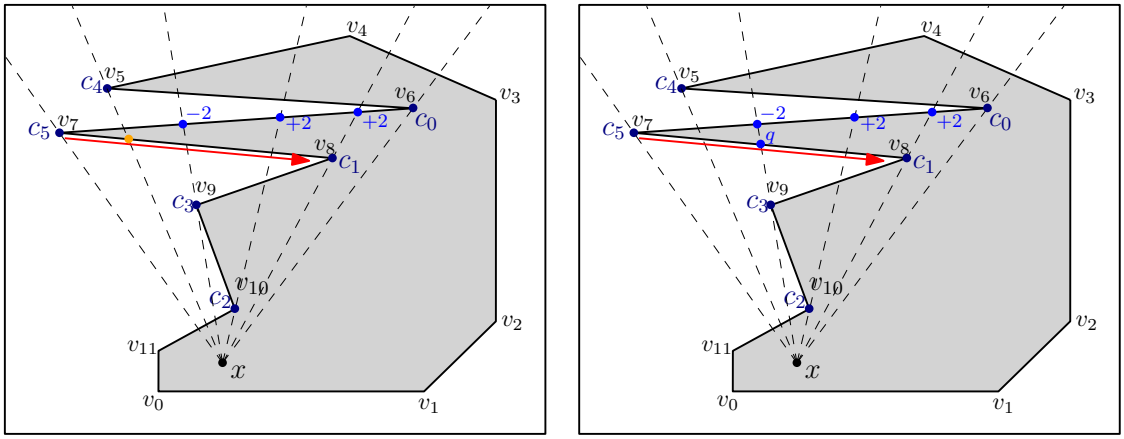
ii) pelo vértice crítico c_{j-1}, caso x esteja do lado direito de v_iv_{i+1} (ver figura 2.26).

Note-se que se deve ir actualizando o valor de j, consoante o vértice crítico que se está a analisar.

Note-se também que é necessário verificar, em relação a cada vértice crítico:

a) se esse vértice crítico está mais próximo de x do que o ponto de intersecção q (ou seja, basta verificar se o vértice crítico se encontra do mesmo lado da aresta percorrida que o ponto x), caso em que se deve etiquetar e guardar o ponto q (ver figuras 2.25b e 2.26b);

b) caso contrário, está-se perante a situação em que o vértice crítico encontra-se mais afastado de x do que o ponto de intersecção q, logo esse ponto q não será utilizado na construção da região de visibilidade, pelo não é necessário etiquetar nem guardar o ponto q (ver exemplo da figura 2.26a).



(a) (b)

Figura 2.26: Passo 3 (caso *ii)*): (a) caso a (ponto laranja não se deve considerar); (b) caso b (etiquetar e guardar ponto q encontrado).

Este procedimento de percorrer a aresta e ir encontrando possíveis pontos de intersecção é feito até se atingir o vértice do outro extremo da aresta percorrida, tendo em conta que, de cada vez que um ponto de intersecção é encontrado, este é imediatamente etiquetado; isto porque, aquando do cálculo deste novo ponto, se dispõe da informação do vértice crítico com o qual está relacionado, informação essa que é necessária para etiquetar o referido ponto.

O procedimento repete-se para todas as arestas do polígono (percorrido em sentido positivo), até se atingir novamente o vértice crítico $c_0 = v_i$ (ver figura 2.27).

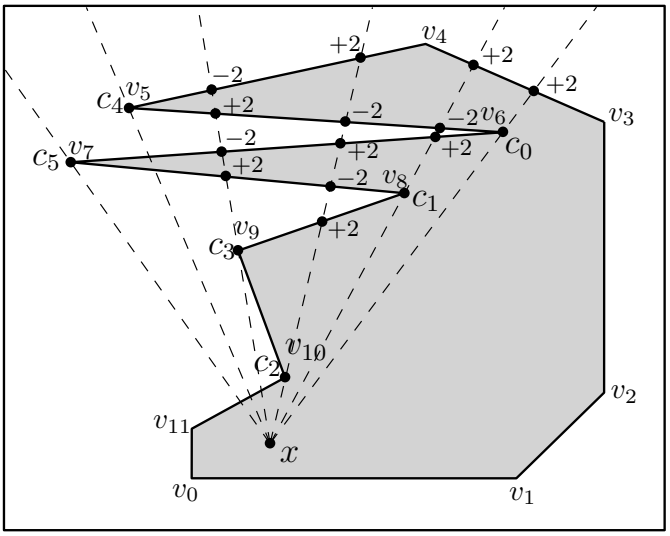


Figura 2.27: Pontos de intersecção etiquetados.

A forma como este passo é feito, representa uma redução significativa no número de operações a executar relativamente ao algoritmo anterior.

Ora, no algoritmo anterior, caso houvesse r vértices críticos, eram feitas exactamente nr operações para encontrar os pontos de intersecção pois ter-se-ia que percorrer todo o polígono (n arestas) por cada vértice crítico em busca desses pontos, o que resulta numa complexidade de $O(n^2)$, assumindo que, no caso assimptótico, há n vértices críticos. Já neste algoritmo, o número de operações é seguidamente apresentado e justificado (assume-se novamente que, de modo assimptótico, há n vértices críticos):

Percorrendo cada aresta do polígono e encontrando os pontos de intersecção da maneira acima descrita, verifica-se que, no máximo, uma aresta terá $n-2$ (excluem-se, obviamente, os vértices extremos da aresta). A aresta com maior número de pontos de intersecção a seguir a esta terá, no máximo, $n-3$ intersecções. Da mesma forma, a aresta seguinte com maior número de intersecções terá $n-4$ pontos a analisar, no máximo. Seguindo este raciocínio, verifica-se que, no máximo, ter-se-á que analisar $\sum_{i=1}^{n-2} i$ pontos no total de todas arestas (obviamente, pelo menos duas arestas não terão intersecções).

Desenvolvendo o somatório apresentado, conclui-se que se terá de executar

$$\sum_{i=1}^{n-2} i = \frac{(n-2)(n-1)}{2} = \frac{n^2-3n+2}{2} \tag{2.3.1}$$

identificações de arestas, o que significa uma redução significativa no número de operações

relativamente aos passos correspondentes do algoritmo anterior. Note-se que o número de operações neste caso é menos de metade que no caso anterior, apesar da complexidade assimptótica idêntica, $O(n^2)$.

Passo 4: Etiquetar segmentos da linha poligonal.

Este passo é feito de igual forma ao do algoritmo anterior, uma vez que todos os vértices críticos e respectivos pontos de intersecção se encontram etiquetados como nesse algoritmo (ver figura 2.28).

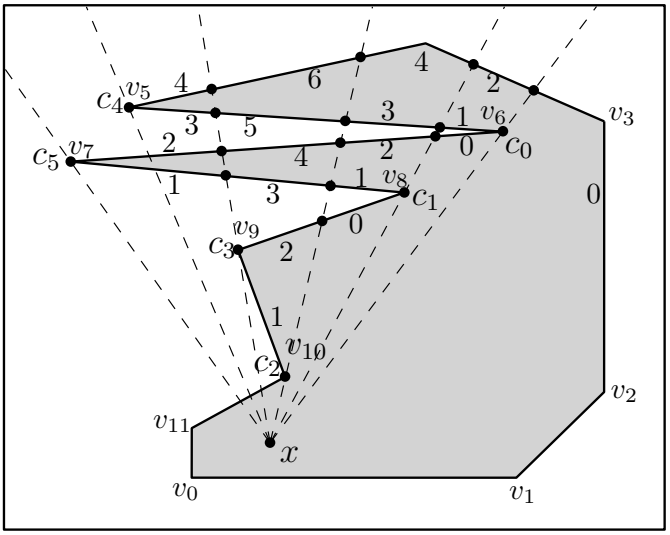


Figura 2.28: Passo 4: segmentos etiquetados.

Tal como no algoritmo anterior, este passo é feito em tempo linear, $O(n)$.

Passo 5: Construir a região de visibilidade.

A fronteira de $Vis_k(x, P)$ é construída ligando, em ordem angular, todos os segmentos com a mesma etiqueta k, usando as semi-rectas incidentes,da mesma forma como no algoritmo anterior (ver exemplo da figura 2.29).

Portanto, a complexidade deste último passo é idêntica ao passo do algoritmo anterior, $O(n)$.

No final, obtém-se uma complexidade algorítmica de $O(n^2)$ mas, como já foi referido, enquanto que no algoritmo anterior esse número de operações pode ser atingido em alguns casos (no passo 2), neste algoritmo melhorado, é necessário de menos de metade das operações, no passo 3 (o único passo que representa complexidade superior a linear).

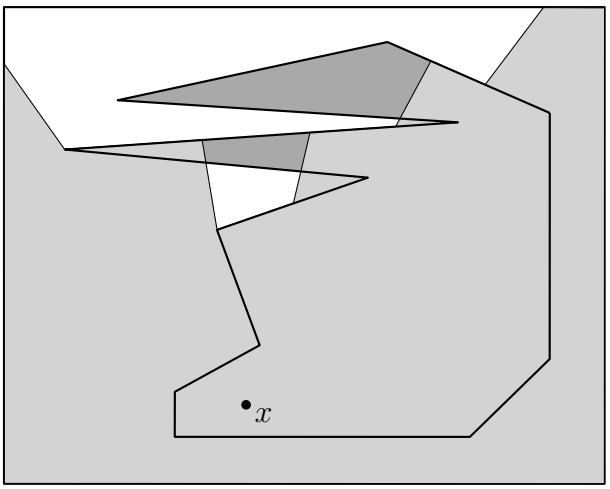


Figura 2.29: Passo 5: Região de visibilidade $Vis_2(x, P)$ a cinzento.

Relativamente aos casos degenerados, pode ser feita um tratamento igual ao descrito para o algoritmo anterior. Mas tal como no algoritmo anterior, na implementação deste algoritmo, não foram tratados os casos degenerados, pelas mesmas razões. No futuro, poder-se-á implementar na aplicação uma função para o tratamento dos casos degenerados que poderá ser invocada tanto na execução de uma algoritmo como noutro, uma vez que o tratamento dos casos degenerados é semelhante.

2.3.2 Algoritmo formalizado

Em seguida, é resumido de maneira formal o Algoritmo de Etiquetagem Melhorado.

Algoritmo 3 Algoritmo de k-cobertura por Etiquetagem Melhorado

Entrada: Polígono P, ponto x e valor k.
Saída: Região de visibilidade $Vis_k(x, P)$.

Passo 1. Encontrar vértices críticos em relação a x e etiquetá-los.

Passo 2. Ordenar os vértices críticos angularmente em torno de x.

Passo 3. Encontrar os pontos de intersecção q das semi-rectas com o polígono e etiquetá-los: Seja v_k o vértice crítico corrente, que corresponde ao vértice $v_i \in V_P, i \in [0, n-1]$. Começa-se a percorrer a fronteira de P (no sentido positivo) a partir de v_k, e executa-se este passo até chegar novamente a v_k. A aresta que está a ser percorrida denota-se por $v_i v_j$.

Passo 3a. Ao percorrer a aresta $v_i v_j$, procurar a primeira intersecção nessa aresta, correspondente à semi-recta com origem em x e que, dependendo da orientação da aresta, passa pelo vértice crítico seguinte v_k ($k = k + 1$, caso $i < j$, ou $k = k - 1$, caso $i > j$). Note-se que é necessário verificar, em relação a cada v_l:

3a.1. se esse vértice crítico está mais próximo de x do que o ponto de intersecção q, etiquetar esse ponto de intersecção.

3a.2. caso contrário, não se considera o ponto de intersecção.

Passo 3b. Executar o passo 3a. até se atingir v_j.

Passo 4. Etiquetar segmentos.

Passo 5. Construção da região de visibilidade $Vis_k(x, P)$.
Ligar, em ordem angular, todos os segmentos com a mesma etiqueta k, usando as semi-rectas incidentes.

2.4 Algoritmo de k-cobertura por Varrimento

O último algoritmo que aqui se apresenta é baseado na ideia de Gregorio Hernández de varrimento do polígono. Por esse motivo, será chamado nesta dissertação de *algoritmo de k-cobertura por varrimento.*

O algoritmo é similar ao algoritmo de varrimento para detectar os pontos de corte de uma família de segmentos no plano, mas adaptado ao problema da determinação da região de k-cobertura. Haverá, então, uma semi-recta de varrimento r com origem em x que varrerá angularmente o polígono, girando em torno de x (ver figura 2.30).

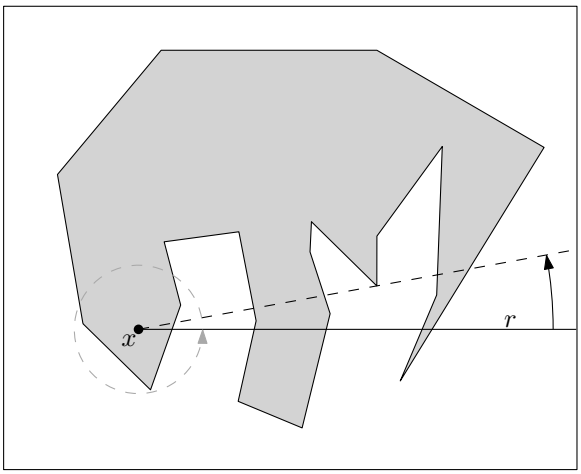


Figura 2.30: Recta de varrimento r.

2.4.1 Descrição do algoritmo

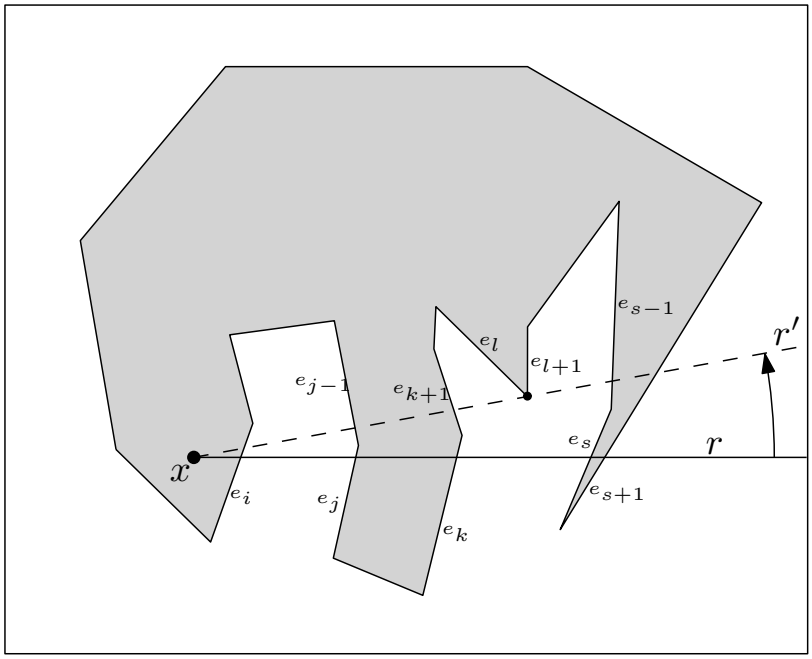
Tal como nos algoritmos anteriores, considera-se um polígono P com n vértices, $V_P = v_0, v_1, ..., v_{n-1}$, ordenados em sentido positivo, e x um ponto do interior de P onde está colocado um k-transmissor. P está contido num rectângulo R e a região de visibilidade $Vis_k(x, P)$ será construída no interior de R. Será necessário ter presente a noção de vértice crítico, apresentada numa secção anterior.

Também se supõe que x está em posição geral, ou seja, não se consideram os seguintes casos degenerados: *i)* x é colinear com dois ou mais vértices críticos de P e *ii)* x pertence a uma recta que passa sobre uma aresta de P.

Associado à semi-recta de varrimento r com origem em x que varrerá angularmente o polígono, girando em torno de x, existirá uma estrutura de dados (uma lista ou array), que se irá actualizando. Esta estrutura da semi-recta r contém a informação de quais lados do polígono que estão a ser intersectados por r naquele momento e será actualizada de cada vez que a semi-recta de varrimento alcançar um vértice crítico, pois haverá que alterar

(actualizando e eliminando ou acrescentando) os lados do polígono intersectados por r (ver figura 2.31).

Na implementação feita deste algoritmo, foi usada uma lista L como estrutura de dados para representar a semi-recta r, lista essa que armazena os pontos de intersecção da semi-recta e as arestas correspondentes. A utilização de listas facilita a inserção e remoção de pontos, sempre que a semi-recta alcança um novo vértice crítico; daí a sua utilização na implementação.



(a)

e_i	e_j	e_k	e_s	e_{s+1}

(b)

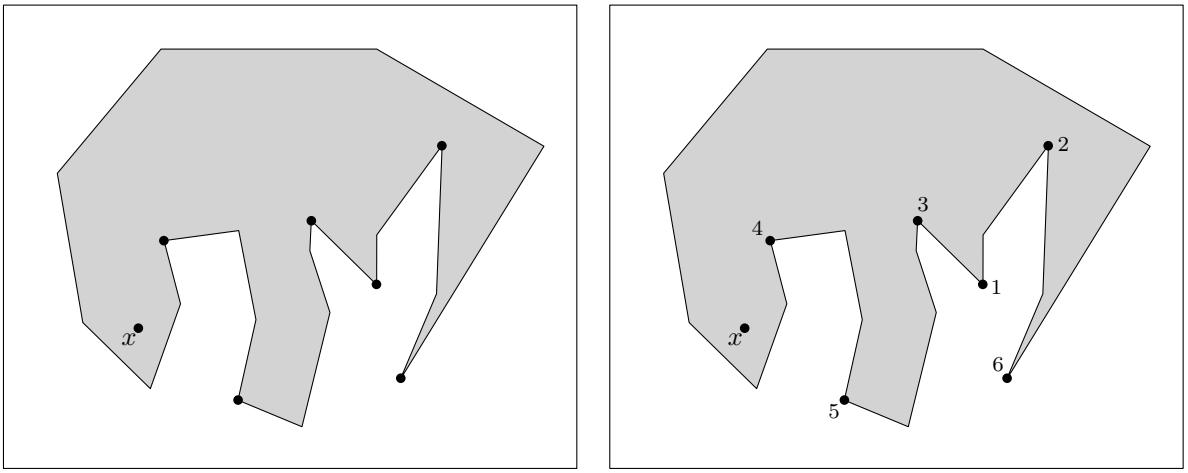
Figura 2.31: (a) Semi-recta r e nova semi-recta (r') que forçará a actualização da estrutura devido ao vértice crítico alcançado; (b) estrutura L associada à semi-recta r com informação das arestas intersectadas.

Os passos do algoritmo são:

Passo 1: Detectar os vértices críticos do polígono em relação a x.

Tal como nos dois algoritmos anteriores, apesar de este se guiar por uma nova estratégia, também há a necessidade de verificar quais os vértices de P que são críticos (ver figura

2.32a). Isto porque estes vértices influenciam mudanças no desenho da fronteira da região de visibilidade por representarem o fim ou o início de obstáculos.



(a) (b)

Figura 2.32: Vértices críticos.

Passo 2: Ordenar angularmente os vértices críticos.

Uma vez que a semi-recta r girará em torno de x e varrerá o polígono, a informação da ordem angular dos vértices críticos será útil para saber qual o vértice crítico seguinte que a semi-recta irá intersectar e onde irá parar para ser actualizada, de modo a ter-se informação sobre por onde é que a fronteira de $Vis_k(x, P)$ prossegue (ver figura 2.32b).

Passo 3: Varrimento.

Este passo faz a construção da fronteira da região de k-cobertura $Vis_k(x, P)$ através do varrimento do polígono, feito por uma semi-recta de varrimento r com origem em x, girando em torno de x.

A semi-recta r de varrimento do polígono será representada, aquando da execução do algoritmo, por uma lista L, que contém a informação de quais os lados que são intersectados por r, em dado momento. Esta lista estará ordenada por ordem crescente de distância do ponto x aos pontos de intersecção dos lados de P que r intersecta ao ponto (ver figura 2.31b). Supõe-se que o índice da lista começa por 0.

Assim, a fronteira de $Vis_k(x, P)$ obtém-se seguindo o lado que ocupa a posição k na lista L que representa a semi-recta de varrimento.

Como os lados nas posições pares da lista têm o interior de P à esquerda e os lados nas posições ímpares da lista têm o interior de P à sua direita, será fácil saber por onde avançar, se se quiser percorrer determinado lado do polígono: em sentido positivo (caso k

seja par) ou negativo (caso k seja ímpar), uma vez que a fronteira de $Vis_k(x, P)$ é sempre construída em sentido positivo.

O processo de varrimento, tem paragens quando a semi-recta r intersecta o vértice crítico seguinte. Nessas alturas, há que actualizar a lista L da seguinte forma (acompanhar a descrição através das figuras 2.31 e 2.33):

Denote-se por r' a nova recta e por r a recta que intersecta o polígono segundo a lista actual L.

Primeiro, *(i)* há que proceder à modificação da lista L, actualizando os lados que r intersecta para os lados que r' intersecta, e depois *(ii)* inserir ou remover o vértice crítico em questão.

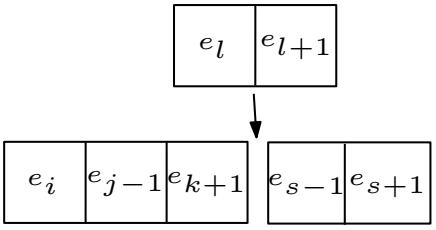


Figura 2.33: Nova lista L correspondente a r'.

Descreve-se em seguida, em mais pormenor os pontos *(i)* e *(ii)*:

i) Para proceder à primeira fase de actualização da lista relativamente à semi-recta r para a semi-recta r', intersecta-se r' com os seguintes lados dos elementos de L que estão nas posições pares, e com os anteriores lados dos elementos de L que estão nas posições ímpares, até alcançar os novos pontos de intersecção, actualizando os elementos de L ao chegar a esses pontos.

ii) Na segunda fase da actualização de L, é necessário introduzir ou remover o vértice crítico que está a ser analisado, consoante este seja do tipo A ou do tipo B (ver figura 2.34):

Tipo A. Os vértices anterior e posterior ao vértice crítico estão à esquerda da semi-recta r. Assim, devem ser removidos os 2 elementos consecutivos da lista correspondentes às arestas incidentes no vértice crítico em questão.

Tipo B. Os vértices anterior e posterior ao vértice crítico estão à direita da semi-recta r. Assim, ao aparecerem duas novas arestas, devem ser inseridas na lista na sua posição correcta, de acordo com a sua distância a x.

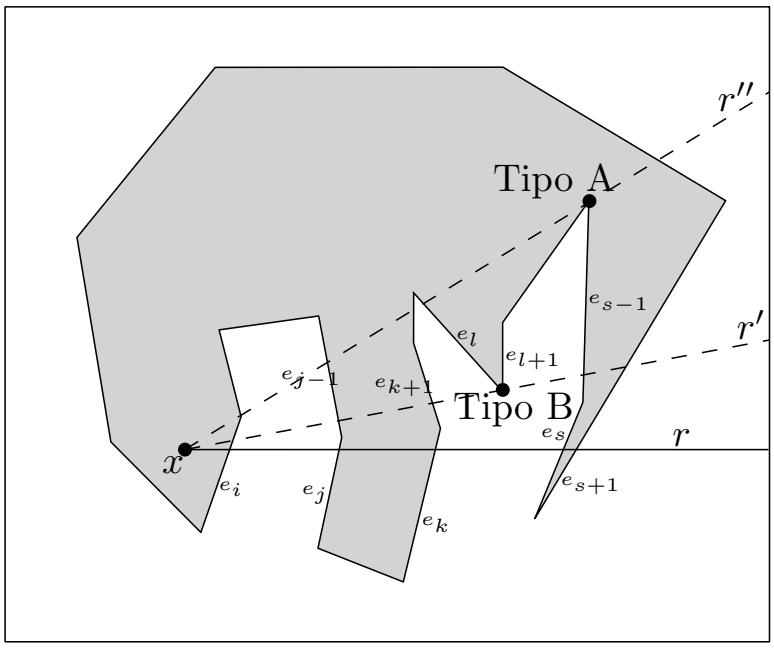


Figura 2.34: Vértices do tipo A e do tipo B.

Note-se que, quer num caso, quer noutro, há necessidade de procurar a posição (índice) de L onde esta modificação (inserção ou remoção) se efectua. Assim, a fronteira de $Vis_k(x, P)$ segue pelo lado que ocupa a posição k na na lista L (ver figura 2.35).

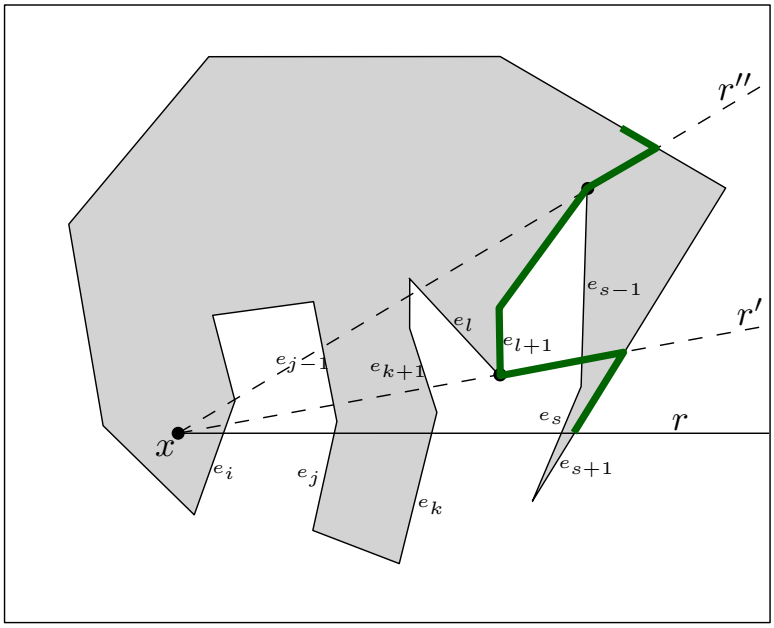


Figura 2.35: Construção da fronteira da região de visibilidade $Vis_4(x, P)$.

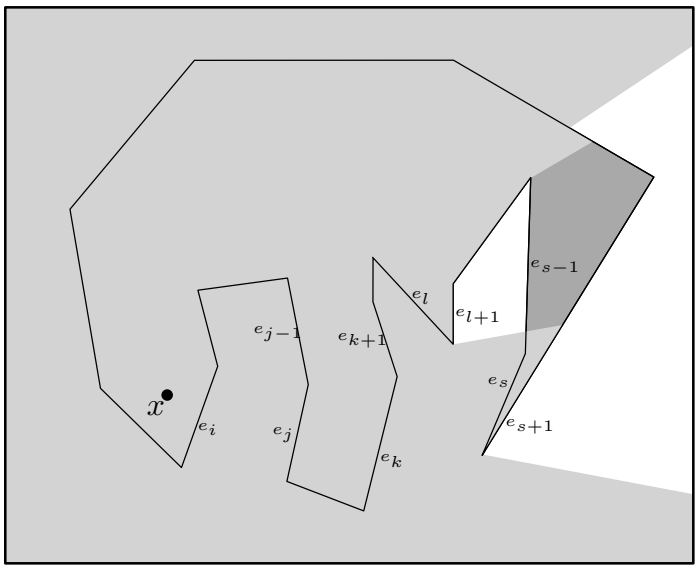


Figura 2.36: $Vis_4(x, P)$.

A análise dos casos degenerados não foi feita para este algoritmo. No entanto, uma vez que os polígonos gerados pela aplicação para a execução deste algoritmo são polígonos aleatórios, a probabilidade de um caso degenerado acontecer é muito reduzida. O tratamento dos casos degenerados ao nível da aplicação desenvolvida é uma questão a tratar no futuro.

2.4.2 Algoritmo formalizado

De seguida, apresenta-se o *algoritmo de k-cobertura por varrimento* resumido de maneira formal.

Algoritmo 4 Algoritmo de k-cobertura por Varrimento

Entrada: Polígono P, ponto x e valor k.
Saída: Região de visibilidade $Vis_k(x, P)$.

Passo 1. Detectar os vértices críticos do polígono em relação a x.

Passo 2. Ordenar angularmente os vértices críticos em relação a x.

Passo 3. Varrimento: Construir a lista L que representa a semi-recta r com origem em x e que passa pelo primeiro vértice crítico, v_0. Executar os seguintes passos, até voltar ao primeiro vértice crítico:

Passo 3a. Procurar o ponto de intersecção da semi-recta r com o lado da posição k da lista L.

3a.1. Se k é par, prosseguir pela fronteira de P em sentido positivo, até encontrar a intersecção da recta r' (que passa pelo vértice crítico seguinte) com a fronteira que se está a percorrer.

3a.2. Se k é ímpar, prosseguir pela fronteira de P em sentido negativo, até encontrar a intersecção da recta r' com a fronteira que se está a percorrer.

Passo 3b. Actualizar a lista L, relativamente ao novo vértice crítico, que se denota por v_i:

3b.1. Intersectar a semi-recta r' com os seguintes lados dos elementos de L que estão nas posições pares, e com os anteriores lados dos elementos de L que estão nas posições ímpares, até alcançar os novos pontos de intersecção, actualizando os elementos de L ao chegar a esses pontos.

3b.1. Se:
(i) os vértices anterior e posterior ao vértice crítico estão do lado direito de $\overrightarrow{xv_i}$ (vértice crítico do tipo A), retirar da lista L os dois lados incidentes nesse vértice crítico (que são elementos consecutivos).
(ii) senão (tipo B), então inserir os dois lados incidentes no vértice crítico na sua posição correcta (de acordo com a sua distância a x) na lista L (serão elementos consecutivos na lista).

2.4.3 Análise da complexidade algorítmica

A análise da complexidade temporal do algoritmo apresentado é feita de seguida, comentando-se o custo de cada passo.

O primeiro passo, de detecção dos vértices críticos de P, tem custo $O(n)$, tal como acontecia em algoritmos anteriores, uma vez que a sua identificação requer que se percorra os n vértices do polígono.

Já o passo seguinte, devido à ordenação dos vértices detectados no passo anterior, tem uma complexidade de $O(nlogn)$.

O terceiro passo, divide-se em duas fases: a actualização da lista que representa a semi-recta de varrimento e a construção da fronteira de $Vis_k(x, P)$. Assim, para cada vértice crítico, pode-se fazer a seguinte análise:

Na primeira fase, a complexidade temporal é de $O(n)$, pois a pesquisa da posição da lista onde se faz a inserção ou remoção do vértice crítico, que está a ser analisado em dado momento, é feita em tempo linear. Isto porque (tendo-se programado o algoritmo em Java, e utilizando listas para representar a recta de varrimento) a pesquisa em listas ordenadas é feita em tempo linear (no caso assimptótico); a utilização de listas (em detrimento de arrays) deve-se ao facto de o comprimento da estrutura ter de variar de acordo com as inserções e remoções que é necessário ir fazendo. Além disso, se considerarmos as actualizações que é necessário fazer em relação aos outros elementos da lista (os quais não estão relacionados com remoções nem inserções), percebe-se que é necessário, para cada vértice crítico, ir actualizando estes elementos, o que resulta mais uma vez em $O(n)$.

A construção da fronteira de $Vis_k(x, P)$, faz-se seguindo o lado que ocupa la posição k na lista que representa a recta de varrimento, pelo que este passo tem complexidade $O(k)$. Uma vez que para $k \geq n$, a região de cobertura corresponderá simplesmente ao rectângulo R, considera-se que este passo pode ser feito em $O(n)$, no limite.

Estas duas fases do terceiro passo resultam, portanto, numa complexidade de $O(n+n)$, pelo que se assume que este passo tem complexidade linear, para cada vértice crítico.

Repetindo este processo para n vértices crítico (valor assimptótico), o terceiro passo apresenta uma complexidade de $O(n^2)$.

Portanto, o *algoritmo de varrimento* apresenta uma complexidade de $O(n^2)$.

Capítulo 3

Desenvolvimento da Aplicação

Com o intuito de testar e comparar o desempenho dos diferentes algoritmos, apresentados na Capítulo 2, e de criar uma ferramenta para futuras pesquisas na área, foi desenvolvida uma aplicação com interface gráfica, onde foram implementados os seguintes algoritmos:

- Algoritmo de k-cobertura por Etiquetagem;

- Algoritmo de k-cobertura por Etiquetagem Melhorado;

- Algoritmo de k-cobertura por Varrimento.

Esta aplicação, desenvolvida utilizando a linguagem Java (versão 1.6) e o ambiente de desenvolvimento NetBeans IDE 6.9.1 [17], recorre a uma outra aplicação anteriormente desenvolvida por A. S. Ramos [6], para a obtenção de polígonos simples aleatoriamente. Assim, a implementação feita ao longo do trabalho integrar também esta última, de modo a que seja gerado um polígono aleatório, escolhido um ponto para fixar um k-transmissor e calculada a região de k-cobertura.

Neste capítulo, serão descritos aspectos importantes da aplicação, assim como da interface gráfica desenvolvida. Por último, serão apresentados alguns resultados experimentais obtidos com recurso a esta aplicação, que visam a comparação da eficiência dos algoritmos implementados.

3.1 Geração Aleatória de Polígonos

Uma vez que a aplicação desenvolvida integra uma outra anterior, será importante explicar em que medida essa aplicação contribui para este trabalho.

Dado que a finalidade da aplicação desenvolvida neste trabalho e a determinação da região de visibilidade $Vis_k(x, P)$, dado um polígono simples P, um ponto x do interior de P representando um k-transmissor e um valor k do número de obstáculos possíveis de serem ultrapassados pelo sinal emitido pelo transmissor sem se perder a qualidade do

sinal, existe a necessidade de obter polígonos simples aleatórios, que permitam testar os algoritmos implementados para a resolução do problema proposto.

O trabalho de A. S. Ramos [6] deu um forte contributo na aplicação desenvolvida para esta dissertação, uma vez que descreve e implementa uma série de heurísticas para o problema de gerar aleatória e uniformemente polígonos simples a partir de um dado conjunto de pontos. Como não é conhecido qualquer algoritmo de complexidade de tempo polinomial, para a geração uniforme de polígonos simples, o trabalho da autora centrou-se em heurísticas que "apesar de não gerarem uma solução óptima para o problema, permitem gerar um número significativo de polígonos simples ou, restringindo o problema de geração aleatória de polígonos simples, gerar apenas polígonos de uma determinada classe, como por exemplo, polígonos estrelados, monótonos ou ortogonais".

A heurística que revelou gerar gama de polígonos simples mais adequada ao tipo de aplicação que se pretendia desenvolver neste trabalho corresponde ao algoritmo **2-Opt Moves** [18]. Isto porque esta heurística permite a geração aleatória de todos os polígonos simples e os polígonos gerados por este algoritmo apresentaram nos testes feitos pela autora uma forma bastante mais "suave" comparativamente aos outros algoritmos implementados no trabalho da autora, que sugeriu assim esta a heurística como "mais adequada para simular uma situação real", como se pretende neste trabalho.

Seria também interessante, aplicar os diferentes algoritmos de determinação da região de k-visibilidade a polígonos ortogonais, uma vez que estes representam uma grande parte da estrutura dos edifícios, mas tal não aconteceu, nomeadamente porque a aplicação desenvolvida no trabalho de A. S. Ramos [6] não contemplava algoritmos para a geração deste tipo de polígonos.

3.1.1 Algoritmo 2-Opt Moves

A heurística 2-Opt Moves, inicialmente idealizada por Zhu et al [18], foi estudada e implementada no trabalho de Auer e Held [19].

É de notar que apenas na biblioteca CGAL [20] foi encontrado um algoritmo disponível para a geração de polígonos simples, que utiliza a estratégia da heurística 2-Opt Moves.

Nesta secção será apresentada a heurística 2-Opt Moves, fornecendo-se os conceitos e notações necessários à sua compreensão, descrito o algoritmo formalmente e analisada a sua complexidade.

3.1.1.1 Descrição do algoritmo

Esta heurística tem como ponto de partida a geração aleatória de um polígono (simples ou não) e posterior remoção das intersecções entre arestas, até que o polígono se torne simples. A geração de um polígono inicial pode ser realizada aplicando uma permutação

dos pontos de um conjunto de pontos S. Para eliminar as intersecções entre arestas não consecutivas, estas são determinadas e guardadas e de seguida aplicam-se os seguintes passos até que tenham sido tratadas todas as intersecções:

1. Escolher aleatoriamente uma intersecção, cujas arestas que lhe dão origem são v_iv_{i+1} e v_jv_{j+1}.
2. Remover do conjunto de intersecções todas as que envolvam pelo menos uma das arestas v_iv_{i+1} e v_jv_{j+i}.
3. Substituir v_iv_{i+1} e v_jv_{j+i} por duas novas arestas v_iv_j e $v_{i+1}vj+1$.
4. Determinar e guardar as intersecções de v_iv_j e de $v_{i+1}v_{j+1}$ com as restantes $n-2$ arestas.

Definição 3.1. *A substituição de duas arestas, v_iv_{i+1} e v_jv_{j+1}, que determinam uma intersecção, por duas novas arestas v_iv_j e de $v_{i+1}v_{j+1}$ é denominada por* 2-opt move *(ver exemplo da figura 3.1).*

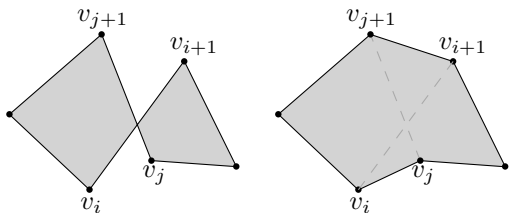


Figura 3.1: Exemplo de um *2-opt move*.

3.1.1.2 Algoritmo formalizado

Na próxima página apresenta-se o algoritmo 2-Opt Moves de maneira formal.

Em relação à notação usada, denota-se por I o conjunto de todas as intersecções entre arestas não consecutivas, por $k = (f_1, f_2)$ a intersecção das arestas f_1 e f_2, que por sua vez correspondem às arestas v_iv_{i+1} e v_jv_{j+1}. Após a execução de um 2-opt move, essas arestas são substituídas por duas novas v_iv_j e $v_{i+1}v_{j+1}$, que se denotam por $e1$ e $e2$, respectivamente.

Algoritmo 5 Algoritmo 2-Opt Moves

Entrada: Conjunto S de n pontos.
Saída: Um polígono simples P, cujos vértices são os pontos de S.

Passo 1. Construir aleatoriamente um polígono P em S, aplicando uma permutação dos pontos de S.

Passo 2. Determinar todas as intersecções de P e guardá-las em I.

Passo 3. Enquanto $I \neq \emptyset$,

Passo 3a. Seleccionar aleatoriamente uma intersecção $j = (f_1, f_2)$ de I;

Passo 3b. Remover de I todas as intersecções de f_1 com qualquer uma das outras arestas de P;

Passo 3c. Remover de I todas as intersecções de f_2 com qualquer uma das outras arestas de P;

Passo 3d. Substituir em P as arestas f_1 e f_2 pelas arestas e_1 e e_2;

Passo 3e. Determinar todas as intersecções entre e_1 e todas as outras arestas de P e adicioná-la a I;

Passo 3f. Determinar todas as intersecções entre e_2 e todas as outras arestas de P e adicioná-la a I.

3.1.1.3 Complexidade e Características da Solução do Algoritmo

Neste algoritmo, é necessário determinar todas as intersecções, o que é feito em tempo $O(n^2)$, uma vez que se verifica cada uma das n arestas com todas as outras.

De seguida, e em cada iteração, que corresponde a um 2-opt move, a escolha aleatória de uma intersecção é feita em tempo constante e a remoção de intersecções leva tempo linear pois, para cada aresta, no máximo, existem n intersecções. A determinação das novas intersecções, após a substituição de arestas, pode também ser efectuada em tempo linear. Assim, para cada iteração é gasto tempo linear.

Como são necessários no máximo $O(n^3)$ 2-opt moves para obter um polígono simples, a complexidade temporal desta heurística é $O(n^4)$.

Esta heurística permite a geração aleatória de todos os polígonos simples, daí que tenha sido escolhido este algoritmo para integrar a aplicação desenvolvida; porém, não o faz de modo uniforme, uma vez que existem polígonos que só podem ser obtidos através de uma única sequência de 2-opt moves e outros que podem ser obtidos através de diferentes sequências de 2-opt moves.

3.2 Implementação

Apresentam-se de seguida alguns aspectos considerados importantes no desenvolvimento da aplicação criada. Desde a estrutura dos dados utilizados na aplicação que facilitaram a implementação dos algoritmos, passando pela implementação propriamente dita, até à estrutura dos ficheiros de entrada e saída, serão explicadas algumas opções tomadas ao longo do desenvolvimento da aplicação e em que medida é que foram aproveitados blocos programados da aplicação anterior.

3.2.1 Estrutura de dados da aplicação

Para uma eficaz implementação dos algoritmos e desenvolvimento da interface gráfica, é imprescindível estruturas de dados adequadas, que facilitem a manipulação dos dados. Nesta secção descrevem-se as classes utilizadas, justificando-se algumas opções tomadas ao longo da implementação.

Uma vez que a aplicação desenvolvida integra uma outra anterior de geração de polígonos (conforme explicado na secção anterior), a maioria das classes foi aproveitada e adaptada à nova aplicação, adicionando-se em alguns casos novos atributos necessários.

Dado que a aplicação manipula constantemente polígonos e conjuntos de pontos, tanto para a geração do polígono como para a construção da região de visibilidade (representada também por um polígono), há necessidade de uma classe que representa os pontos e que se denominou por **Ponto** (ver figura 3.2).

Cada instância desta classe é caracterizada por uma abcissa x e uma ordenada y, ambos valores reais, que representam as coordenadas de determinado ponto na área desenho da interface gráfica.

A classe Ponto tem também um atributo *ângulo* (também valor real), devido à necessidade de guardar o valor do ângulo entre a semi-recta horizontal com origem no transmissor e que se estende para a direita e a semi-recta também com origem no transmissor e um determinado vértice crítico, aquando da ordenação angular destes vértices executada pelos três algoritmos implementados.

Além disso, esta classe tem um objecto *seguinte* da classe Ponto, que faz referência ao ponto imediatamente a seguir, no caso de um objecto da classe Ponto fazer parte de uma **Lista** (classe descrita mais adiante), como é o caso dos vértices de um polígono.

Estes atributos até agora apresentados, são os atributos presentes na classe Ponto da aplicação anterior. No entanto, outros atributos foram acrescentados a esta classe. Além disso, a maioria das funções criadas anteriormente para a manipulação desta classe (por exemplo, o cálculo da distância entre dois pontos, o cálculo da área do triângulo formado por três pontos) foi mantida, pela sua necessidade, e outras novas foram implementadas.

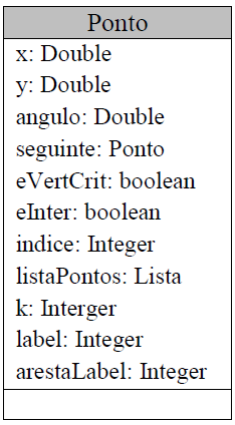


Figura 3.2: Classe Ponto e seus atributos.

À classe Ponto foram acrescentados dois atributos booleanos, *eVertCrit* e *eInter*, que informam, respectivamente, se determinado ponto é ou não um vértice crítico do polígono em relação ao transmissor e se é ou não um ponto de intersecção das arestas do polígono com uma semi-recta referente a determinado vértice crítico (isto é, que passa por um vértice crítico) com origem no transmissor. Assim, ao percorrer o polígono, é possível verificar quais os pontos do polígono que são ou não vértices críticos e quais o pontos que são ou não pontos de intersecção.

Cada instância Ponto tem ainda um atribuo *indice*, que é um valor inteiro que representa o índice da ordenação angular, no caso de o ponto ser um vértice crítico.

No caso de determinado ponto ser um vértice crítico, há também a necessidade de saber quais os pontos de intersecção a si associados, isto é, na execução dos algoritmos deve ser possível aceder-se aos pontos de intersecção das arestas do polígono com a semi-recta com origem no transmissor e que passa pelo vértice crítico em causa. Nesse sentido, a solução criada foi incluir na classe Ponto um atributo denominado *listaPontos*, que é uma Lista (classe a seguir descrita) que contém os pontos de intersecção associados a um ponto que seja vértice crítico (caso um ponto não seja vértice crítico a lista está vazia). Esta *listaPontos* é ainda útil numa outra situação, em que este atributo representa uma lista de pontos com um significado diferente: é o caso de uma instância Ponto representar um transmissor. Como pode haver vários transmissores num mesmo polígono, há a necessidade de associar a cada ponto transmissor uma lista dos pontos que representam a sua região de visibilidade (depois de determinada).

Ainda no caso de um ponto representar um transmissor, existe nesta classe um atributo k, que é um valor inteiro que representa o valor máximo de k obstáculos (paredes/arestas) que o sinal do transmissor consegue atravessar.

Por último, na classe Ponto encontram-se os atributos *label* e *arestaLabel*, necessários para memorizar, respectivamente, a etiqueta dos vértices e pontos de intersecção ao longo da fronteira do polígono e a etiqueta de determinado segmento, aquando da execução dos algoritmos de etiquetagem. Optou-se por representar um segmento pelo ponto de origem do segmento, ponto esse, cujo atributo *seguinte* é o outro extremo do segmento. Assim,

um segmento fica perfeitamente definido na implementação e o valor da sua etiqueta é guardado no atributo *arestaLabel* do ponto origem do segmento (ponto que surge primeiro quando se percorre o polígono positivamente).

A necessidade de processar pontos como um conjunto único é indispensável na manipulação de polígonos e na construção da região de k-visibilidade. Deste modo, incluiu-se a classe **Lista** (já presente na aplicação anterior, agora com algumas adaptações) que representa uma lista ligada linear, cujos elementos são pontos (ver figura 3.3a). Esta classe é composta por uma instância da classe Ponto, que indica o primeiro elemento da lista e por um valor inteiro *tamanho* correspondente ao número de elementos da lista. Assim, a classe Lista, para além de ser utilizada para representar o conjunto de pontos inicial a partir dos quais se gera um polígono, também permite a representação de polígonos e cadeias poligonais e implementa alguns passos de alguns algoritmos, relativos à manipulação de vértices de um polígono, como por exemplo a etiquetagem de pontos e segmentos nos algoritmos de etiquetagem e a construção da semi-recta de varrimento no algoritmo por varrimento.

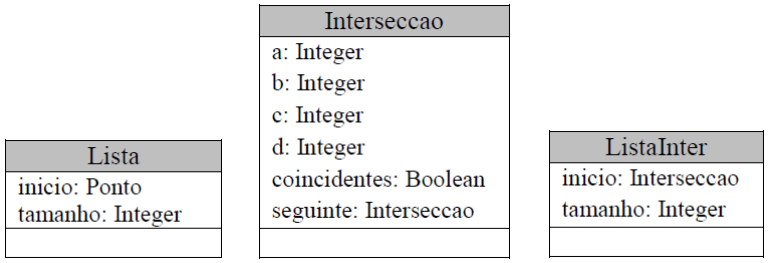


Figura 3.3: Classes e respectivos atributos: (a) Classe Lista; (b) Classe Interseccao; (c) Classe ListaInter.

Já a classe **Interseccao** (ver figura 3.3b), apenas tem utilidade no âmbito da heurística 2-Opt Moves, para permitir guardar todas as intersecções entre arestas do polígono. Cada instância desta classe é caracterizada por quatro valores inteiros que representam os índices (a localização na lista de pontos que define o polígono) dos vértices do polígono que são extremos das arestas que se intersectam. Para além disso, possui um atributo booleano, que indica se as arestas em questão são coincidentes ou não, e um objecto da classe Interseccao, que permite aceder à intersecção seguinte.

Uma vez que na maioria das vezes o número de intersecções é consideravelmente elevado, definiu-se a classe **ListaInter** (ver figura 3.3c), também apenas utilizada no âmbito da heurística 2-Opt Moves. Esta classe possibilita o armazenamento de todas as intersecções e é especificada por apenas dois atributos, um objecto da classe Interseccao, que permite aceder à primeira intersecção da lista, e um valor inteiro, que determina o número de elementos da lista de intersecções.

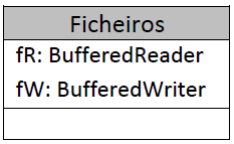


Figura 3.4: Classes Ficheiros e seus atributos.

Por último, foi também mantida a classe denominada *Ficheiros* (apenas acrescentando e alterando algumas funções relativas à manipulação deste tipo de estrutura) que, como o seu nome indica, auxilia na manipulação de ficheiros (ver figura 3.4). As instâncias desta classe permitem guardar informação relativa a um conjunto de pontos de entrada, a polígonos gerados, a transmissores colocados no interior de polígonos e às respectivas regiões de visibilidade, possibilitando também a leitura deste tipo de dados a partir de um dado ficheiro. Cada objecto é caracterizado por um ficheiro de escrita e outro de leitura, consistindo cada um deles numa instância da classe BufferedReader e da classe BufferedWrite (classes predefinidas de Java), respectivamente.

3.2.2 Implementação dos algoritmos e interface gráfica

Nesta subsecção apresentam-se alguns aspectos gerais da implementação, que facilitaram o desenvolvimento da aplicação.

A implementação dos algoritmos para a determinação da região de k-visibilidade foi feita desenvolvendo algoritmos auxiliares para cada passo. Isto permite com que alguns destes algoritmos auxiliares possam ser invocados em vários dos algoritmos de k-visibilidade, como por exemplo o algoritmo de determinação dos vértices críticos, processo descrito no capítulo anterior.

Em relação à interface gráfica, o seu aspecto assemelha-se ao da aplicação anterior, agora com mais opções pois, além da geração de polígonos, é possível colocar transmissores e determinar a região de k-visibilidade do transmissor. Houve necessidade também de implementar outros algoritmos relacionados com a interface gráfica. Por exemplo, sendo possível mover um transmissor colocado dentro de um dado polígono, criou-se um algoritmo para detectar o transmissor mais proximo do ponto onde o utilizador clica (num raio máximo pré-definido), de modo a seleccionar esse transmissor para a deslocação de acordo com o movimento que o utilizador faz. Mias aspectos relativamente à interface gráfica serão dados na secção seguinte.

3.2.3 Estrutura dos ficheiros de Entrada/Saída

Tal como na aplicação anterior, esta nova permite guardar e aceder a ficheiros que contêm dados relativos ao conjuntos inicial de pontos, polígono formado por esses pontos, localização e valor de k de cada k-transmissor e respectivas regiões de k-visibilidade determinadas pelos algoritmos implementados.

Estes dados são armazenados num ficheiro de texto, segundo algumas regras, para poderem ser lidos pela aplicação que poderá desenhar os objectos relativos à informação desse ficheiro. Foi gerado um exemplo de um ficheiro (ver figura 3.5), com os dados de um polígono com seis vértices, que foi gerado pela aplicação, um 2-transmissor colocado dentro do polígono e sua região de 2-visibilidade determinada também pela aplicação. Cada ficheiro de texto pode ser constituído por 3 secções, no máximo: uma referente ao conjunto de pontos (dados de entrada), outra relativa aos vértices polígono gerado pela aplicação e uma outra referente aos transmissores e suas às regiões de visibilidade. Claro que existe a possibilidade de um ficheiro não ter toda esta informação se, por exemplo, o utilizador optar por guardar os dados antes de a aplicação determinar a região de visibilidade dos transmissores colocados; nesse caso, o ficheiro terá apenas 3 secções. A forma como a primeira e segunda secções se organizam foram mantidas da aplicação anterior e a nova secção foi feita com base na forma como a estrutura destes ficheiros foi inicialmente pensada. Os ficheiros apresentam as coordenadas e toda a informação necessária no caso de o utilizador der indicações à aplicação para voltar a desenhar esses objectos.

```
##############################################
#   Ficheiro de dados para o Polygon Generator   #
##############################################
#  Vértices  #
6
281.0 483.0
299.0 272.0
8.0 102.0
252.0 5.0
213.0 186.0
488.0 34.0
#  Polígono  #
281.0 483.0
488.0 34.0
299.0 272.0
252.0 5.0
213.0 186.0
8.0 102.0
#  Modems  e Regiões #
1
100.0 180.0 2
10
364.34807878592187 302.2111721020342
413.0277612112584 196.62094307316414
284.52999431228875 189.79805279534276
252.0 5.0
256.34285714285716 0.0
0.0 0.0
0.0 500.0
500.0 500.0
500.0 364.9246231155779
364.34807878592187 302.2111721020342
```

Figura 3.5: Exemplo de ficheiro.

Neste tipo de ficheiros cada ponto é especificado numa linha através das suas coordenadas, sendo que primeiro vem o valor da abcissa, imediatamente seguido de um espaço, que por sua vez é seguido pelo valor da ordenada.

A primeira secção contém uma primeira linha que apresenta um inteiro, indicando o número de elementos do conjunto de vértices inicial. Seguidamente, apresenta uma listagem das coordenadas desses pontos, um em cada linha.

A secção relativa ao polígono, consiste numa listagem dos seus vértices (pontos anteriores) pela ordem em que aparecem no polígono.

Na terceira secção, a primeira linha contém um valor inteiro que indica o número de modems colocados pelo utilizador. Se de facto houver modems colocados, então aparecerá um outra linha com as coordenadas do transmissor seguido de um espaço e um valor inteiro representando o valor k desse transmissor. Se o utilizador tiver dado indicação para aplicação determinar a região de visibilidade desse transmissor, então o ficheiro apresentará uma linha com o número de vértices da fronteira da região de k-visibilidade desse transmissor e nas linhas seguintes uma listagem dos vértices dessa fronteira. Segue-se depois a informação relativa aos outros transmissores colocados no polígono gerado (se os houver), informação essa que se estrutura no ficheiro de forma semelhante à descrita para o primeiro transmissor.

3.3 A Interface Gráfica

Com o objectivo de facilitar a interacção da aplicação com o utilizador, simplificando a introdução de dados de entrada e a visualização dos resultados, foi criada uma interface gráfica para a aplicação, a que se chamou "Aplicação para a determinação da região de visibilidade". Nas figuras 3.6 e 3.7 observa-se o aspecto geral da interface gráfica desenvolvida.

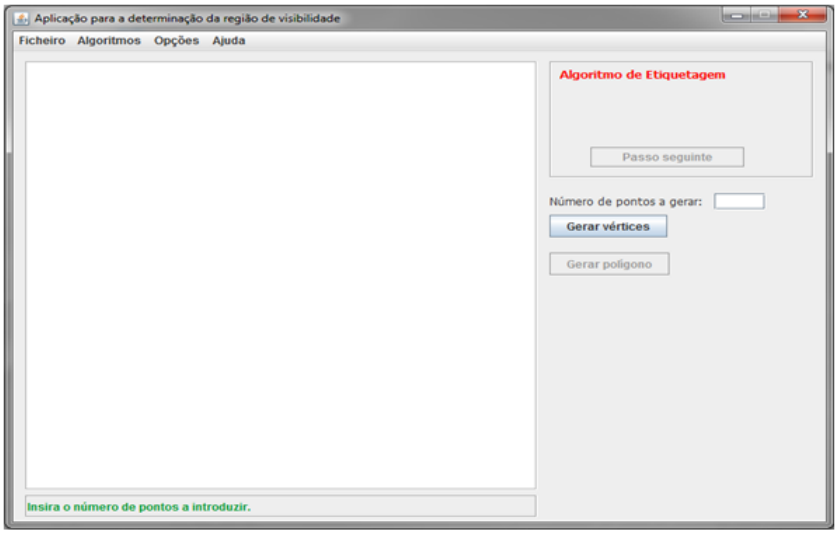


Figura 3.6: Aspecto geral da aplicação quando iniciada.

A interface gráfica da aplicação conta com uma barra de menus (na parte superior), uma tela onde são desenhados os dados introduzidos e os resultados, uma área com botões e com indicações dos processos que estão a decorrer ao longo da utilização da aplicação (do lado direito) e uma área de comentários (na parte inferior) onde são dadas algumas indicações ao utilizador sobre como proceder em determinado momento.

Como se pode observar, inicialmente é necessário dar indicações para a geração de um polígono (ver figura 3.6). Essas indicações têm a ver com os pontos que servirão de vértices na geração de um polígono, que é feita (depois de introduzidos os pontos) clicando no botão "Gerar polígono". Após aparecer o polígono determinado pela aplicação, deverá ser introduzido pelo utilizador um transmissor, clicando na área de desenho no sítio onde pretende que este fique situado. O transmissor deverá ser colocado dentro do polígono, caso contrário aparecerá um aviso na área de comentários da parte inferior pois, conforme a especificação dos algoritmos no capítulo anterior, a região de visibilidade é calculada para transmissores situados no interior de um dado polígono e foi assim que foram implementados. Após a introdução de um transmissor, o utilizador deverá introduzir o respectivo valor k e pode então indicar à aplicação que determine a respectiva região de visibilidade clicando no botão colocado do lado direito para essa finalidade (ver figura 3.7). Portanto, o aspecto da aplicação vai sendo modificado à medida que o utilizador interage com ela, adaptando-se a cada momento, com novas opções que vão aparecendo na interface ou que vão sendo desactivadas (do lado direito da aplicação). O utilizador pode, se quiser, colocar vários transmissores num mesmo polígono, sendo que os respectivos valores k são colocados junto do ponto que representa o transmissor, na área de desenho. Isto permitirá estudar quantos transmissores, com determinados valores k e em determinada localização, são necessários para cobrir todo o polígono.

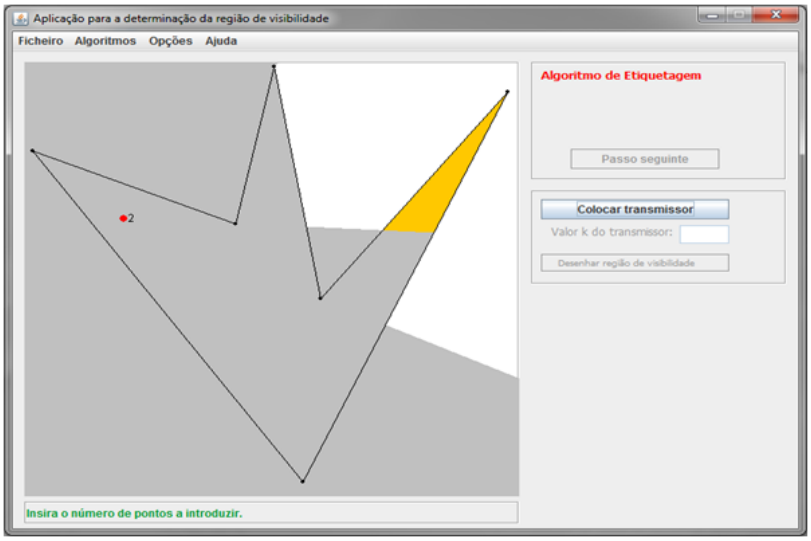
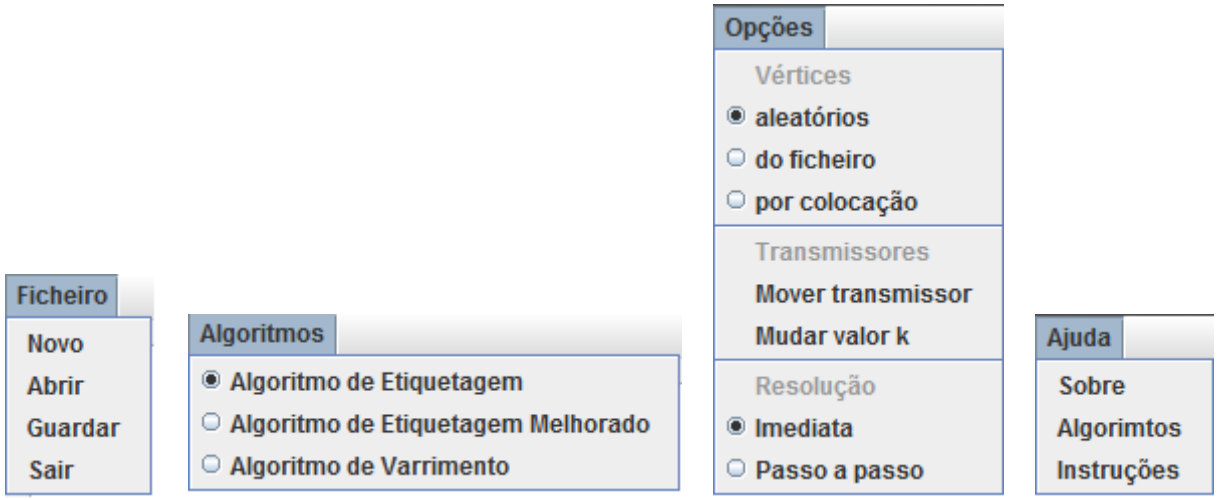


Figura 3.7: Aspecto geral da aplicação com os dados do ficheiro da figura 3.5.

A barra de menus contém um conjunto de várias opções, que a seguir se explicam.

No menu, designado Ficheiro, é possível escolher quatro opções (ver figura 3.8a): a opção *Novo* permite limpar todos os dados e iniciar uma nova introdução de dados, *Abrir* permite ao utilizador aceder aos dados de um determinado ficheiro que serão assim desenhados na tela, *Guardar* possibilita armazenar os dados introduzidos na tela num ficheiro e a opção *Sair* fecha a aplicação.

No menu Algoritmos (ver figura 3.8b), é possível escolher, de entre três, o algoritmo que se pretende aplicar para a determinação da região de visibilidade de um dado transmissor colocado num polígono. De acordo com o algoritmo seleccionado, é apresentado o seu nome na área de descrição de algoritmos no canto superior direito (ver figura 3.7).



(a) (b) (c) (d)

Figura 3.8: Barra de menus: (a) Menu *Ficheiros*; (b) Menu *Algoritmos*; (c) Menu *Opções*; (d) Menu *Ajuda*.

O menu Opções (ver figura 3.8c) apresenta três tipos de opções.

A primeira refere-se à forma como são introduzidos os vértices (dados de entrada) do polígono a ser gerado: *i)* através da leitura de pontos a partir de um ficheiro, *ii)* através da aplicação que gera de forma aleatória um número de pontos (vértices) indicados pelo utilizador ou *iii)* através de pontos fornecidos directamente pelo utilizador, clicando na janela de visualização. Existe uma área, no canto inferior direito, onde são apresentadas as coordenadas de localização do cursor na tela de visualização, o que poderá auxiliar o utilizador na colocação de pontos.

O segundo tipo de opção refere-se a possibilidades de acção sobre transmissores. Dado um transmissor colocado num polígono, é possível movê-lo sobre o polígono. Para isso, dado que podem existir vários transmissores num polígono, o utilizador deverá (conforme

indicado na área de comentários, na parte inferior) clicar sobre o transmissor que se pretende mover. Será seleccionado o transmissor mais próximo do ponto clicado, até distância máxima pré-definida. O utilizador deverá manter o botão do rato pressionado sobre o transmissor e movê-lo dentro da área do polígono, permitindo esta opção ir visualizando as alterações na região de visibilidade. Esta opção pode ser particularmente útil se se estiver interessado em perceber qual(quais) o(s) ponto(s) do polígono onde deve ser preferencialmente colocado um dado transmissor de forma a que o seu sinal abranja a maior área possível. Existe ainda uma outra opção relativa à alteração do valor k de um dado transmissor. Para isso o utilizador deve clicar sobre o transmissor cujo valor k pretende modificar (pois, tal como explicado acima, poderão existir vários transmissores colocados num polígono), depois de seleccionado o ponto (a área de comentários na parte inferior informe quando o ponto tiver sido seleccionado) deverá alterar-se o campo relativo ao valor k (no lado direito da aplicação). Isto permite também perceber quais as alterações na região de visibilidade de um transmissor (clicando no botão "gerar região de visibilidade", depois de alterado o valor k), consoante a potência do seu sinal (representada pela valor k).

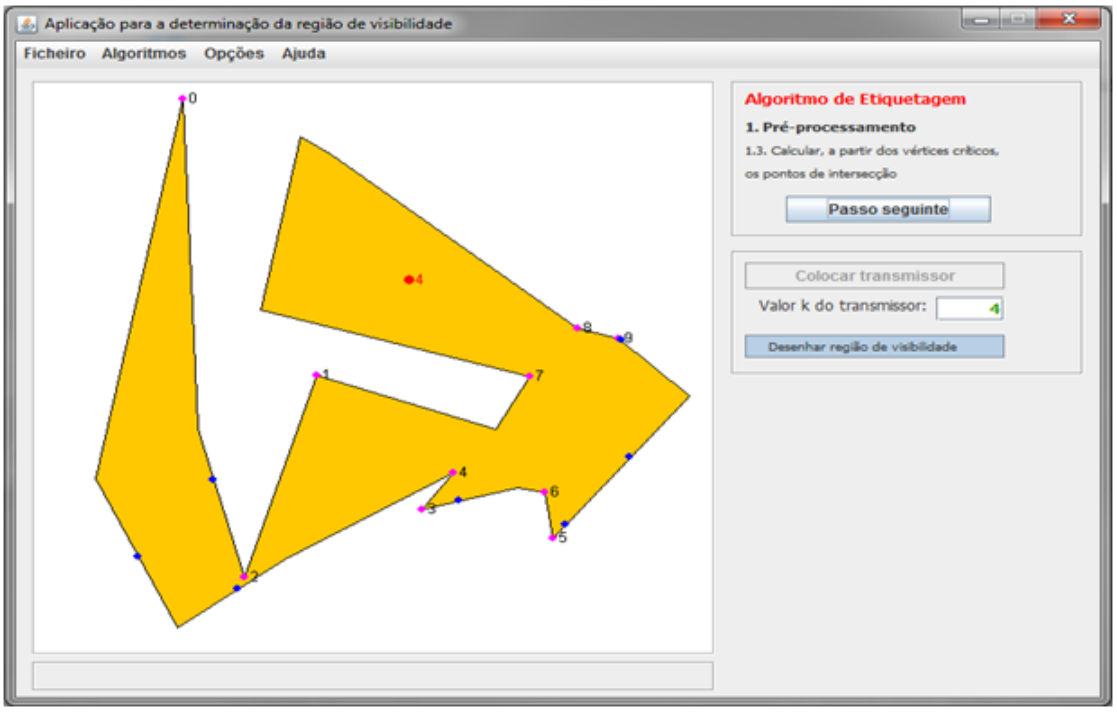


Figura 3.9: Exemplo de execução passo a passo.

Relativamente às opções de forma de resolução (isto é, como deverão ser executados os algoritmos de determinação da região de visibilidade), o utilizador poderá escolher que a resolução seja realizada de forma imediata, isto é, o algoritmo é executado até ao fim podendo apenas observar-se o seu resultado final (a região de visibilidade gerada) (ver figura 3.7). Em alternativa, e visto que é um dos objectivos desta interface ajudar na compreensão

geral dos algoritmos implementados, pode-se escolher a opção "Passo a Passo" que permite acompanhar e observar passo a passo a execução de cada algoritmo. Ao seleccionar esta opção, o utilizador deve indicar, clicando no botão criado para esse efeito, o começo da determinação da região de visibilidade, o que fará com que seja executado o primeiro passo, com os resultados a aparecerem na tela e uma descrição desse passo na área de descrição dos algoritmos no canto superior direito. O utilizador deverá depois ir clicando no botão "Passo seguinte", para visualizar a continuação do algoritmo. Por vezes, são também apresentados os sub-passos, nos casos em que os passos são compostos por várias operações.

Na figura 3.9, é apresentada a execução de um dos passos do algoritmo de etiquetagem: os vértices críticos já foram detectados (pontos rosa) e ordenados angularmente em torno do transmissor (de acordo com a numeração colocado junto a cada um deles) e, de acordo com a legenda da descrição do algoritmo (no canto superior direito), no passo apresentado foram calculados os pontos de intersecção (a azul).

Da barra de menus, faz ainda parte o menu Ajuda, ilustrado na figura 3.8d, que disponibiliza informação sobre a aplicação, sobre os algoritmos implementados (ver figura 3.10) e uma série de instruções que facilitam a manipulação da interface gráfica (ver figura 3.11).

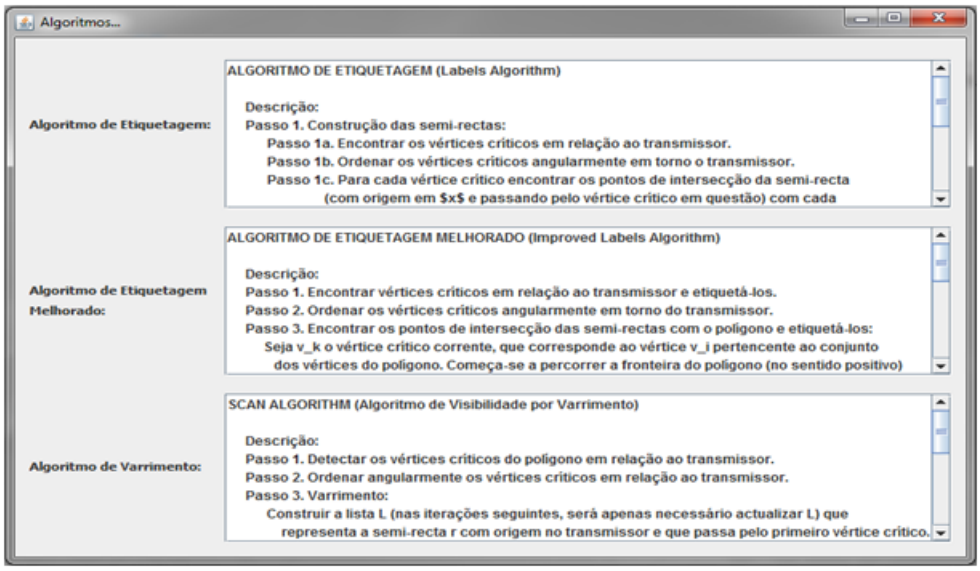


Figura 3.10: Opção *Algoritmos* do menu *Ajuda*.

Seguidamente apresentam-se também alguns exemplos da determinação da região de k-cobertura pela aplicação desenvolvida. Estes exemplos (figuras 3.12, 3.13 e 3.14), foram alguns dos resultados obtidos nos testes feitos na próxima secção (de resultados experimentais).

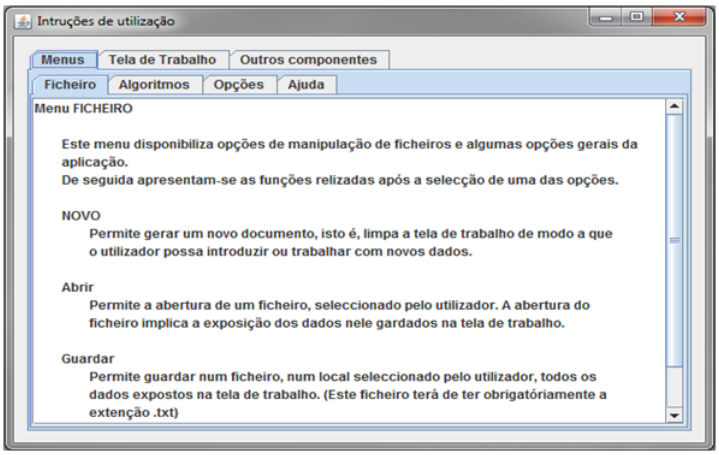


Figura 3.11: Opção *Algoritmos* do menu *Ajuda*.

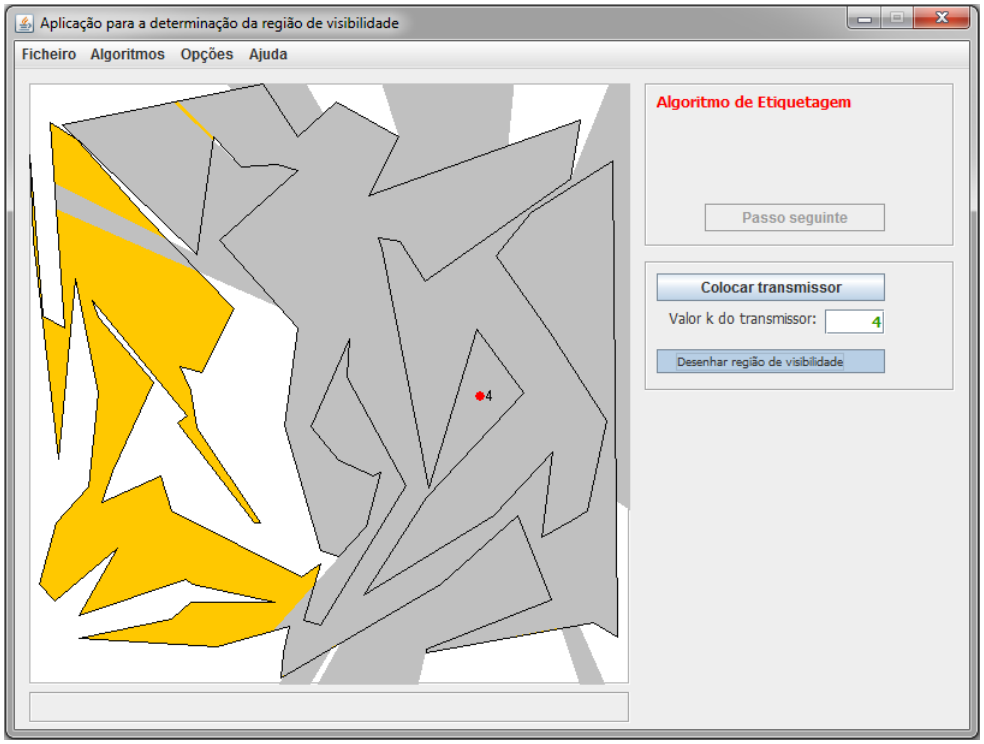


Figura 3.12: Exemplo da região de cobertura determinada pela aplicação de um 4-transmissor colocado num polígono de 100 vértices.

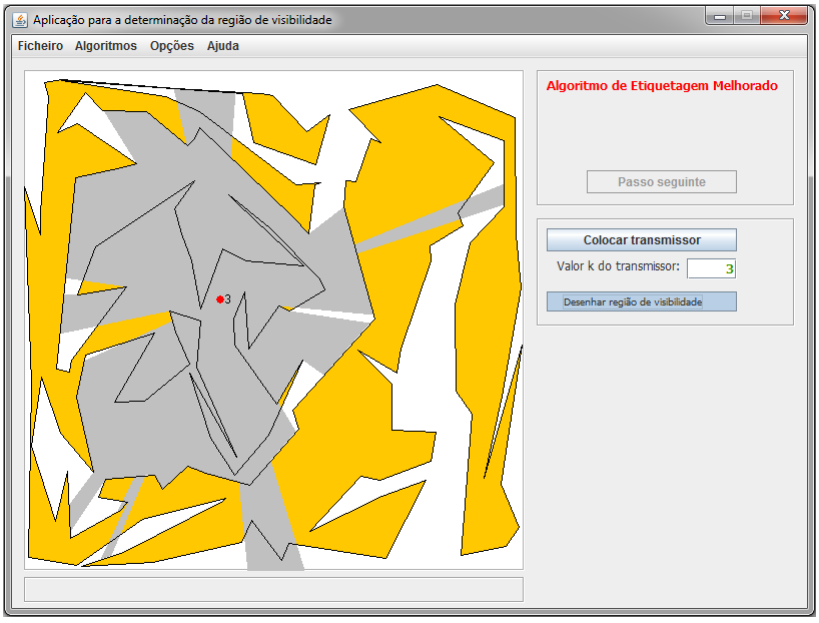


Figura 3.13: Exemplo da região de cobertura determinada pela aplicação de um 3-transmissor colocado num polígono de 150 vértices.

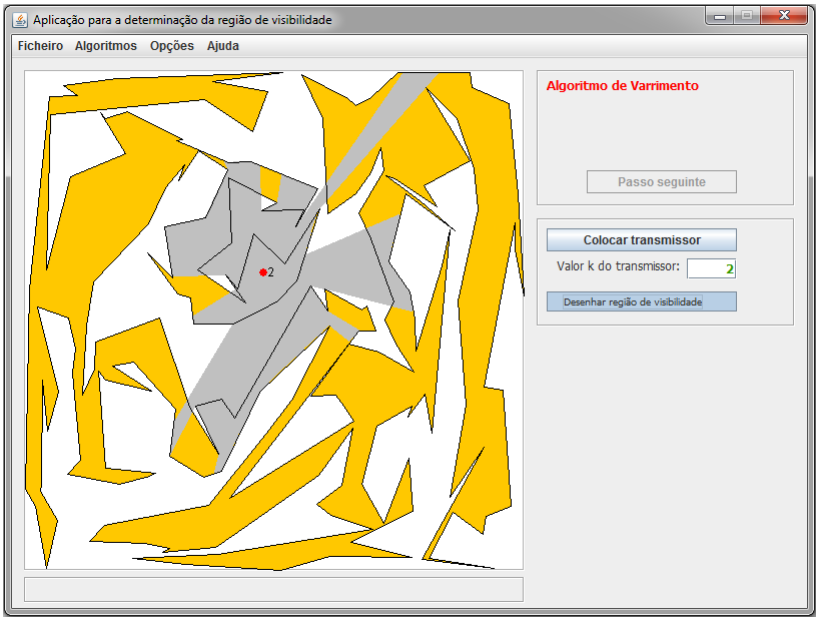


Figura 3.14: Exemplo da região de cobertura determinada pela aplicação de um 2-transmissor colocado num polígono de 200 vértices.

3.4 Resultados Experimentais

De seguida, apresentam-se alguns resultados obtidos experimentalmente, utilizando a aplicação desenvolvida. Os resultados obtidos avaliam os tempos de execução de cada um dos algoritmos, de modo a fazer-se uma análise comparativa.

Os testes foram realizados num computador com processador Intel®Core^{TM} I5 M520 a 2.40 GHz e com 4 GB de memória RAM, num sistema operativo de 64 bits, em ambiente Windows7 Professional.

A análise dos tempos de execução de cada um dos algoritmos, foi efectuada aplicando cada um deles a polígonos de 10, 25, 50, 75, 100, 125, 150, 175 e 200 vértices com um k-transmissor colocado, fazendo-se também variar a sua potência para $k = 2, 3$ e 4. Dado um determinado número de vértices e um dado valor k, cada algoritmo foi executado 10 vezes, gerando-se um polígono diferente em cada execução e colocando o transmissor no seu interior; assim os valores apresentados nas tabelas referem-se à média do tempo de execução. É importante referir também que, gerado um polígono e colocado um transmissor no seu interior, o mesmo polígono foi usado para fazer a medição de cada um dos 3 algoritmos, fazendo variar também o valor de k ($k = 2, 3, 4$), em cada um dos algoritmos. Do tempo de execução contado para cada um dos algoritmos, não faz parte o tempo necessário para a geração de um polígono aleatório com determinado número de vértices.

Na tabela seguinte (3.1) apresentam-se as médias do tempo de execução de cada algoritmos, de acordo com o número de vértices do polígono usado e com o valor k do transmissor nele colocado. Para simplificar, o *algoritmo de etiquetagem* é referido, nas tabelas e nos gráficos que se seguem, como *alg1*, o *algoritmo de etiquetagem melhorado* é mencionado como *alg2* e o *algoritmo de varrimento* como *alg3*.

	$k = 2$			$k = 3$			$k = 4$		
	Tempo (milisseg)			Tempo (milisseg)			Tempo (milisseg)		
n	alg1	alg2	alg3	alg1	alg2	alg3	alg1	alg2	alg3
10	0,69	0,45	0,47	0,61	0,44	0,49	0,66	0,43	0,50
25	2,11	0,94	1,18	1,65	0,98	1,14	1,60	1,00	1,10
50	4,65	1,25	1,42	3,70	1,25	1,41	3,72	1,25	1,42
75	6,32	0,97	1,75	8,69	1,07	1,70	10,98	2,18	1,69
100	14,41	2,01	2,87	15,12	2,15	2,79	16,16	2,22	2,65
125	20,11	2,31	2,89	18,40	2,30	3,08	18,09	2,34	3,46
150	29,50	3,35	4,68	25,14	3,43	4,68	27,49	4,02	4,19
175	32,25	3,36	4,34	28,59	3,83	4,89	28,45	4,13	4,29
200	55,09	6,01	7,53	43,75	7,39	6,61	41,05	6,24	6,70

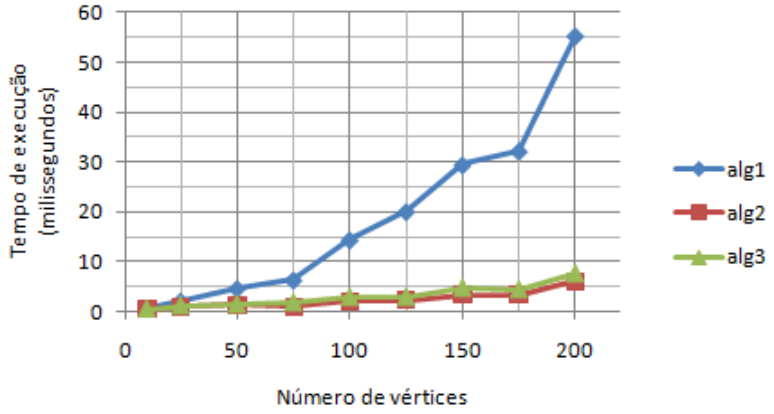
Tabela 3.1: Tempos médios de execução de cada algoritmo.

Observa-se que, fazendo variar o valor de k ($k = 2, 3, 4$), o tempo de execução de cada algoritmo é semelhante. No entanto, as diferenças no tempo de execução entre o Algoritmo de Etiquetagem e os outros dois (o Algoritmo de Etiquetagem Melhorado e o Algoritmo de Varrimento) são visíveis (ver figura 3.15). Apesar de os três algoritmos terem complexidade $\mathcal{O}(n^2)$, o Algoritmo de Etiquetagem tem um tempo de execução maior, tal como esperado. Nos três gráficos apresentados na figura 3.15, principalmente em 3.15a, a linha que representa o tempo de execução do Algoritmo de Etiquetagem assemelha-se ao desenho de uma função quadrática. No caso dos outros dois algoritmos, o aumento do tempo de execução, à medida que o número de vértices do polígono gerado aumenta, não é tão significativo; mas é espectável que, para valores mais altos do número de vértices, o tempo de execução reflicta de forma mais visível a complexidade de $\mathcal{O}(n^2)$ deste dois algoritmos. No entanto, o tempo de pré-processamento (geração de um polígono aleatório pela aplicação) necessário à realização de um estudo para valores mais elevados do número de vértices era elevado, pelo que só se registaram valores para um número de vértices até 200.

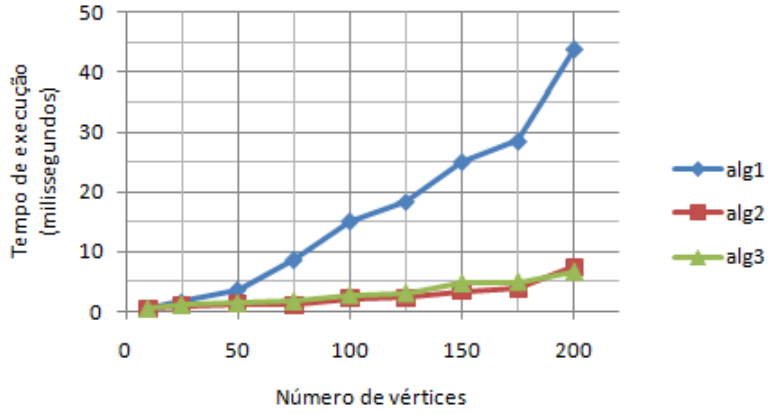
Note-se que, no caso do Algoritmo de Etiquetagem, quanto maior o número de vértices do polígono aleatório gerado para a realização destas medições, maior é o desvio-padrão (ver tabela 3.2), enquanto que no outros dois algoritmos o desvio-padrão é quase sempre menor que 1 (existindo apenas um pequeno aumento de acordo com o aumento do número de vértices). Isto porque se verificou uma grande discrepância nos valores obtidos nas medições feitas no caso do Algoritmo de Etiquetagem, para valores do número de vértices mais elevados, sendo que, por vezes, os valores mais baixos do tempo de execução deste algoritmo eram semelhantes aos valores obtidos para os outros dois algoritmos. Portanto, nem sempre o Algoritmo de Etiquetagem apresenta um comportamento muito diferente dos outros dois algoritmos avaliados, ao contrário do que sugerem os gráficos da figura 3.15.

	Desvio padrão do tempo de execução (milissegundos)								
	$k = 2$			$k = 3$			$k = 4$		
n	alg1	alg2	alg3	alg1	alg2	alg3	alg1	alg2	alg3
10	0,30	0,08	0,13	0,25	0,16	0,19	0,21	0,18	0,14
25	0,70	0,18	0,39	0,61	0,22	0,25	0,41	0,17	0,24
50	1,20	0,31	0,24	0,71	0,20	0,38	1,14	0,19	0,44
75	2,64	0,29	0,20	3,67	0,56	0,13	4,78	2,08	0,15
100	5,77	0,45	0,40	6,86	0,28	0,71	7,49	0,51	0,35
125	9,10	0,77	0,73	7,65	0,84	0,75	5,78	0,16	0,72
150	13,99	1,44	2,28	12,13	0,84	1,62	16,06	1,16	0,64
175	18,76	0,89	1,46	19,28	1,22	1,36	20,08	0,70	0,63
200	39,12	0,90	1,65	44,36	3,98	1,89	43,53	0,89	0,85

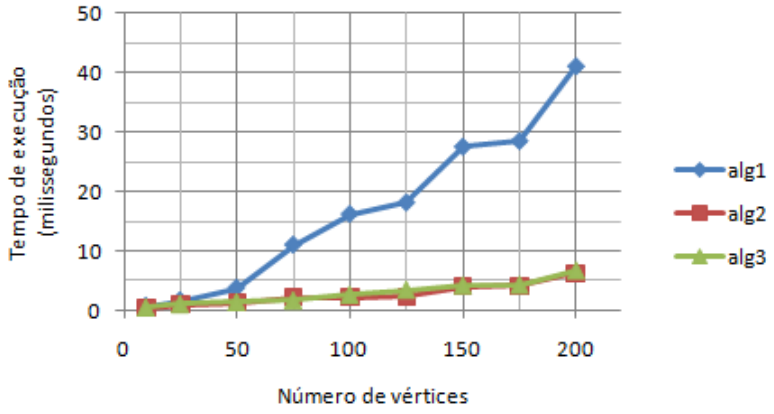
Tabela 3.2: Desvio padrão dos tempos de execução de cada algoritmo.



(a)



(b)



(c)

Figura 3.15: Relação entre os tempos médios de execução de cada algoritmo e o número de vértices do polígono gerado no qual se colocou um k-transmissor: (a) $k = 2$; (b) $k = 3$; (c) $k = 4$.

Capítulo 4

Conclusão

Esta dissertação abordou o problema da determinação da região de cobertura de um k-transmissor colocado num polígono simples. Apresentaram-se os resultados existentes na área e, devido à preocupação em encontrar soluções mais eficientes do que algoritmo já existente, o *algoritmo de k-cobertura por etiquetagem*, foram descritos dois novos algoritmos para a resolução do problema: o *algoritmo de k-cobertura por etiquetagem melhorado* e o *algoritmo de k-cobertura por varrimento*. Estes dois novos algoritmos propostos, apesar de terem a mesma complexidade assimptótica que o já existente, $\mathcal{O}(n^2)$, apresentaram estratégias diferentes para a resolução do problema, o que torna preferível a sua execução em detrimento do algoritmo anterior.

De forma a avaliar experimentalmente a suspeita de que a execução dos dois novos algoritmos era mais rápida e devido à necessidade da existência de uma aplicação para a resolução do problema para futuros estudos, foram implementados os três algoritmos referidos. Para tal, recorreu-se, em parte, a uma aplicação já existente para a geração de polígonos aleatórios, integrando-a na nova aplicação, que conta também com uma interface gráfica.

Na análise dos resultados experimentais, de facto, os dois novos algoritmos propostos revelaram-se mais eficazes, apresentando um tempo de execução mais baixo do que o algoritmo de etiquetagem existente anteriormente.

Terminada esta dissertação, crê-se que será difícil desenvolver novos algoritmos para a resolução do problema estudado que apresentem complexidade menor que $\mathcal{O}(n^2)$, uma vez que se pretende determinar a região de cobertura de um k-transmissor, qualquer que seja o valor de k colocado num polígono simples de n lados.

Assim, de futuro, faz sentido determinar novos algoritmos para a construção da região de k-visibilidade, para um valor fixo de k, de modo a conseguir-se reduzir a complexidade para, por exemplo, $\mathcal{O}(nlogn)$. A existência de um algoritmo para a determinação da região de 2-visibilidade em tempo $\mathcal{O}(nlogn)$ seria importante, não só porque tal algoritmo teria uma complexidade menor do que os algoritmos apresentados nesta dissertação, mas também porque se verifica que a situação de ter um 2-transmissor corresponde a muitos casos na realidade.

Na aplicação desenvolvida, a maior preocupação foi para a implementação dos algoritmos apresentados (além da integração da aplicação anterior de geração aleatória de polígonos), tendo sido depois acrescentadas várias operações (como, por exemplo, a opção de deslocar o dado k-transmissor ou mudar o seu valor k). No entanto, outras operações poderão ser implementadas no futuro e acrescentadas à aplicação desenvolvida, de forma a aumentar as possibilidades de escolha para um utilizador, como por exemplo, a opção de eliminar um dado transmissor já colocado num polígono, entre muitas outras.

Também seria importante que a análise dos casos degenerados, referida no Capítulo 2, fosse alvo de uma análise mais profunda, com vista à sua implementação e adição à aplicação desenvolvida, de modo a cobrir todos os casos possíveis e tornar a aplicação o mais correcta e completa possível.

Uma vez que o problema do número mínimo de k-transmissores ainda se mantém em aberto, faz sentido a utilização da aplicação para o estudo de novas soluções.

Além disso, dado que não foi possível encontrar qualquer software, que possibilite a determinação da região de cobertura de um k-transmissor colocado num polígono simples, seria um contributo, em trabalho futuro, a disponibilização na Internet da aplicação desenvolvida.

Em trabalho futuro, será também importante a abordagem do problema da determinação da região de cobertura de um k-transmissor, adicionando novos factores ao problema, de modo a aproximar cada vez mais o problema à realidade. O facto de a espessura e o material das paredes (arestas) influenciar a qualidade do sinal do k-transmissor após atravessar esses obstáculos e o facto de o alcance do sinal na realidade não ser infinito devem ser considerados em futuras abordagens.

Bibliografia

[1] R. Honsberger. *Mathematical Gems II.* Mathematical Association of America, 1976. {1,8}

[2] V. Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory Series B 18*, pages 39–41, 1975. {1,8}

[3] A. K. Lin D. T. Lee. Computational complexity of art gallery problems. pages 276–282, March 1986. {2}

[4] D. Flores-Peñaloza T. Hackl C. Huemer J Urrutia B. Vogtenhuber O. Aichholzer, R. Fabila-Monroy. Modem illumination of monotone polygons. *In Proc. 25th European Workshop on Computational Geometry EuroCG '09, Brussels, Belgium*, pages 167–170, 2009. {2,9,10}

[5] A. M. Martins. *Geometric Optimization on Visibility Problems: Metaheuristic and Exact Solutions.* PhD thesis, Departamento de Matemática, Universidade de Aveiro, 2009. {2,7,10,17,27,33}

[6] A. S. Ramos. Métodos heurísticos para a geração de polígonos simples. Master's thesis, Departamento de Matemática, Universidade de Aveiro. {7,8,55,56}

[7] D. Lee. Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing 22(2)*, pages 207–221, 1983. {8,18,19}

[8] J. O'Rourke. *Art gallery theorems and algorithms.* Oxford University Press, New York, NY, USA, 1987. {8}

[9] S. Ghosh. *Visibility Algorithms in the Plane.* Cambridge University Press, New York, NY, USA, 2007. {8}

[10] J. Urrutia R. Fabila-Monroy, A. Vargas. On modem illumination problems. *XIII Spanish Workshop on Computational Geometry*, 2009. {10}

[11] A. K. Lin D. T. Lee. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory 32(2)*, pages 276–282, 1986. {10}

[12] A. Aggarwal. *The art gallery theorem: its variations, applications and algorithmic aspects.* PhD thesis, 1984. {10}

[13] S. Canales G. Hernández M. Abellanas, E. Alba. Solving the illumination problem with heuristics. *In: T. Boyanov, S. Dimova, K. Georgiev, G. Nikolov (eds) Numerical Methods and Applications, Springer, Lecture Notes in Computer Science, vol 4310*, pages 205–213, 2006. {10}

[14] Benedikt L. Davis, M. *Computacional models of space: Isovists and isovist fields.* ComputerGraphics ans Image Processing, 11, 1979. {18,19}

[15] D. Avis H. ElGindy. *A linear algorithm for computing the visibility polygon from a point.* Journal of Algorithms, 2, 1981. {18}

[16] R. B. Simpson B. Joe. Corrections to lee's visibility polygon algorithm. *BIT, 27*, pages 458–473, 1987. {19}

[17] Netbeans ide 6.1, http://www.netbeans.org/. {55}

[18] J. Snoeyink J.S.B. Mitchell C. Zhu, G. Sundaram. Generating random polygons with given vertices. *Computational Geometry: Theory and Applications, 6(5)*, pages 277–290, 1996. {56}

[19] M. Held T. Auer. Rpg - heuristics for the generation of random polygons. *Journal of Graphics Tools*, November 1996. {56}

[20] `CGAL` - computational geometry algorithms library, http://www.cgal.org/. {56}